FACHHOCHSCHULE DORTMUND

TECHNISCHE HOCHSCHULE KÖLN

MASTERTHESIS

Sentiment Analyse von informellen Kurztexten im Unternehmenskontext

Abschlussarbeit zur Erlangung des Mastergrades Master of Science in der Fachrichtung Wirtschaftsinformatik

Autor:
Sebastian HAEP
Matrikel-Nr.:
7096113
Verbundstudiengang:
Wirtschaftsinformatik
Telefon:
0177/9723936
E-Mail:
sebastian.haep@mail.de

Erstprüfer:
Prof. Dr. Heide FAESKORN-WOYKE
Zweitprüfer:
Prof. Dr. Jan KARPE

Vorgelegt am: 30. Mai 2017

Fachhochschule Dortmund
University of Applied Sciences and Arts

Technology Arts Sciences
TH Köln

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

CEO	**C**hief **E**xecutive **O**fficer
PB	**P**eta**b**yte
GB	**G**iga**b**yte
KB	**K**ilo**b**yte
MB	**M**ega**b**yte
API	**A**pplication **P**rogramming **I**nterface
REST	**Re**presentational **S**tate **T**ransfer
ACID	**A**tomicity, **C**onsistent, **I**solation, **D**urability
BASE	**B**asically **A**vailable, **S**oft State, **E**ventually Consistent
XML	E**x**tensible **M**arkup **L**anguage
JSON	**J**ava**S**cript **O**bject **N**otation
HTML	**H**yper**t**ext **M**arkup **L**anguage
DDL	**D**ata **D**efinition **L**anguage
NLP	**N**atural **L**anguage **P**rocessing

Kapitel 1

Einleitung

„Data are becoming the new raw material of business.“[1]

Craig Mundie, Microsoft

Das Zitat von Craig Mundie, dem Berater von Microsofts Chief Executive Officer (CEO) trifft sehr gut, wie wichtig Daten in der heutigen Zeit für Unternehmen sind. Sie können nicht nur viel über die Kunden verraten, sondern können zudem eine Basis für neue Geschäftsmodelle sein. Daten dienen auch dazu Wissen zu generieren, um bessere Entscheidungen treffen zu können, wodurch Unternehmen gewissermaßen einen Wettbewerbsvorteil erlangen können. Um eine Entscheidung auf Basis von generiertem Wissen zu tätigen, werden Informationen benötigt. Diese Informationen sind nicht einfach vorhanden, sondern müssen zuerst generiert werden. Die Datenflut, die durch das Internet und soziale Medien entsteht[2], bietet Unternehmen eine immer weiter wachsende Quelle aus Daten von und über Kunden. Der Kunde gibt im Internet viel über sich und sein Verhalten preis. Bei den Daten handelt es sich vorrangig um unstrukturierte Daten wie Bilder, Texte oder Videos. Im Laufe der Zeit haben sich viele Techniken, Methoden und Software-Werkzeuge etabliert, um diese Daten immer besser analysieren zu können. Besonders im Bezug auf Marketing, Kundenzufriedenheit und Kundenservice bieten sich viele Chancen für Unternehmen.

1.1 Problemstellung und Zielsetzung der Arbeit

Für viele Unternehmen ist die Analyse solcher unstrukturierter Daten Neuland und wenig greifbar, wie *Klein* in ihrem Artikel anhand von Statistiken

[1] Economist und SAP, 2010, S. 2.
[2] Vgl. Haluschak, 2016.

zur Nutzung von Big Data Lösungen aufzeigt. Demnach nutzen beispielsweise nur 14% der Handelsunternehmen entsprechende Lösungen zur Analyse.[3] Daher sind auch die Herausforderung und der Nutzen, die mit der Analyse der Daten einhergehen, für Unternehmen oft nur schwer vorstellbar. Öffentliche Bewertungen in sozialen Medien bringen eine Transparenz der Leistungsqualität von Unternehmen mit sich. Da sich Nutzer häufiger bei Unzufriedenheit öffentlich an Unternehmen wenden, ist das Risko eines Imageschadens sehr groß. Daraus ergibt sich für Unternehmen die Notwendigkeit von kurzfristigen, kompetenten Reaktionen auf die Kundenanfragen, um die Kundenzufriedenheit wiederherzustellen. Doch wie sollen Unternehmen die Flut an Informationen und Daten schnellstmöglich aufarbeiten, um den Anforderungen ihrer Kunden gerecht zu werden? Einen möglichen Ansatz dafür liefert die vorliegende Masterthesis. Diese Arbeit soll anhand eines praktischen Beispiels mit Hilfe der Sentiment Analyse zeigen wie Unternehmen die Daten nutzen können.

1.2 Vorgehensweise

Die Arbeit ist dabei in sechs Kapitel gegliedert. Nach dem Einleitungskapitel soll in Kapitel 2 der Begriff Big Data erörtert und näher auf Hadoop, als Big-Data-Plattform, eingegangen werden. Darauffolgend werden in Kapitel 3 die Grundlagen des Data Minings behandelt, da diese auch die Grundlage für das sogenannte Text Mining bilden, zu dem auch die Sentiment Analyse gehört. Die Sentiment Analyse und ihre verschiedenen Methodiken und Ansätze werden in Kapitel 4 erklärt. Nach dem theoretischen Teil der Arbeit folgt in Kapitel 5 das praktische Beispiel,anhand von Amazons Kundendienst auf Twitter, wie eine Sentiment Analyse im Unternehmenskontext eingesetzt werden kann. Dabei wird unter anderem auf die Twitter API, auf den Entwurf der Architektur, als auch auf die Umsetzung der Sentiment Analyse eingegangen. In Kapitel 6 wird ein abschließendes Fazit gezogen und ein Ausblick gegeben.

[3] Vgl. Klein, o. J.

Kapitel 2

Big Data

2.1 Anforderungen

„There were 5 exabytes of information created between the dawn of civilization through 2003, but that much information is now created every 2 days, and the pace is increasing."[4]

Eric Schmidt, Executive Chairman bei Google in 2010

Vor allem in den letzten Jahren ist der Begriff „Big Data" durch die Medien sehr verbreitet worden und es entstand ein regelrechter Hype. Gerade in den Medien wird Big Data und dessen Einsatz immer wieder sehr stark vereinfacht dargestellt. Mit Big Data gehen auch Anforderungen, bzw. Herausforderungen an ein solches System einher. Allgemein wird dies häufig durch die drei, beziehungsweise vier V's dargestellt.[5]

- **Volume**: Die naheliegendste Herausforderung, die mit Big Data einhergeht, ist das hohe Datenaufkommen. Einer Studie von EMC zufolge nimmt das weltweite Datenaufkommen sehr schnell zu. Der Studie nach, welche im Jahr 2014 veröffentlicht wurde, wächst das weltweite Datenvolumen bis zum Jahr 2020 um den Faktor 10.[6] Das weltweite Datenvolumen soll sich, der Studie nach, jährlich verdoppeln. Dies geschieht vor allem durch den Vormarsch des Internet of Things, der Vernetzung von Sensoren und beispielsweise auch Haushaltsgeräten. Gerade dadurch erhöht sich das Datenaufkommen enorm. Gleiches gilt auch für Daten die im Social Media-Umfeld generiert werden. Beispielsweise nutzen täglich 1,13 Milliarden Menschen Facebook und

[4] Upbin, 2012.
[5] Vgl. Ohlhorst, 2012, S. 4.
[6] EMC, 2014.

auf Twitter werden pro Tag rund 500 Millionen Tweets gepostet. Wenn Unternehmen Big-Data-Lösungen implementieren, können sehr schnell viele Terabyte oder auch Petabyte anfallen. Die Herausforderung dabei besteht dann darin, diese Daten zu sammeln, dauerhaft zu speichern und der Möglichkeit sie weiter zu verarbeiten.

- **Variety**: Unter Variety versteht man die unterschiedlichen Arten von Daten, die im Big-Data-Umfeld gesammelt und weiterverarbeitet werden können. Hier wird in erster Linie vor allem zwischen strukturierten und unstrukturierten Daten unterschieden. Die unstrukturierten Daten sind gerade deswegen interessant, da sie bisher meist nicht analysiert und weiterverarbeitet wurden, beziehungsweise werden konnten. Zu den unstrukturierten Daten zählen E-Mails, Texte, Audiodateien, Videos, Bilder, Tweets, Facebookposts oder ähnliches. Alleine auf Instagram, einem sozialen Netzwerk für Bilder, werden täglich rund 95 Millionen Bilder hochgeladen. Hierbei ermöglichen Big-Data-Lösungen auch unstrukturierte Daten zu analysieren und diese weiterzuverarbeiten.

- **Velocity**: Mit Velocity ist die Geschwindigkeit, mit der die Daten generiert werden, gemeint. Das beste Beispiel hierfür sind gerade Sensordaten. Sensoren können in sehr kurzen Zeitintervallen, Werte ausgeben, die dann gespeichert und verarbeitet werden müssen. Ein weiteres Beispiel für die Geschwindigkeit mit der Daten produziert werden, sind Facebook-Likes. Pro Sekunde wird die Like-Funktion in Facebook rund 52.000 mal genutzt, ausgehend von einer Gesamtzahl an Likes pro Tag von 4,5 Milliarden. Je nach Datenart und Anwendungsfall müssen Daten in Echtzeit oder zumindest zeitnah zu Informationen weiterverarbeitet werden können.Dies erfordert eine sofortige Speicherung und Weiterverarbeitung der Daten, schon bei deren Entstehung. Zudem müssen die Daten dauerhaft gespeichert werden.[7]

- **Veracity**: Zusätzlich zu den drei bisher genannten V's wird in der Literatur, mit Veracity, häufig ein viertes V genannt.[8] Vor allem die ersten drei V's sind mit Hilfe der auf dem Markt verfügbaren Systeme und Methoden gut zu handhaben. Mit Veracity kommt eine Herausforderung für Big Data dazu, die nicht einfach durch Software erledigt werden kann. Bei Veracity handelt es sich um die Glaubhaftigkeit

[7] Vgl. Ohlhorst, 2012, S. 5.
[8] Vgl. Ohlhorst, 2012; Hurwitz, 2013; Buyya, Calheiros und Dastjerdi, 2016.

der Daten, die gespeichert und verarbeitet werden sollen. IBM war einer der ersten, die Veracity als vierte Dimension der Herausforderungen im Zusammenhang definierte.[9] Aus IBM's Infografik geht auch hervor, dass zweidrittel der Manager den Daten, auf deren Basis sie Entscheidungen treffen, nicht vertrauen. Jedoch sind die Ursachen für die Unglaubwürdigkeit der Daten nicht genau definiert. Die beiden Wissenschaftlerinnen Victoria Rubin und Tatiana Lukoianova haben in einem wissenschaftlichen Artikel diesen Umstand genauer betrachtet und stellen mögliche Ursachen fest.[10] Die Ursachen sind ihrer Meinung nach die "[...] subjectivity, deception and implausibility [...]"[11], also dass Daten nicht objektiv genug, nicht richtig, beziehungsweise betrügerisch und nicht plausibel sind.

2.2 Anwendungsfälle

„*Big data is not about the data.*"[12]

Gary King, Harvard University

Gary King von der Harvard University drückt mit dem oben aufgeführten Zitat genau das aus, was häufig unter dem Begriff Big Data missverstanden wird. Bei Big Data geht es nicht ausschließlich um die Daten selbst oder deren schiere Menge. King versucht damit auszudrücken, dass es bei Big Data vor allem um die Analyse der Daten geht. Die Analyse dieser immerzu wachsenden Datenmengen und auch der Weiterentwicklung von Analysemethoden hilft dabei, bessere Erkenntnisse aus den Daten ziehen zu können. Big-Data-Lösungen bieten die Möglichkeit, bisher ungelöste oder nur schwer lösbare Problemstellungen lösen zu können, aufgrund neuer Analyseverfahren, besserer Technik und Systemen. Gerade große Unternehmen wie Google, Amazon und Facebook sind dafür bekannt, intensiv auf Big Data zurückzugreifen. *Ohlhorst* beschreibt in seinem Buch *Big Data Analytics: Turning Big Data Into Big Money*, dass beispielsweise Amazon jede Information über den Kunden sammelt. Dazu gehören Suchanfragen und die Einkäufe der Kunden. Mit Hilfe von Big Data kann Amazon dem Kunden so sehr präzise Produkte vorschlagen, die er kaufen sollte. Dies erfolgt durch

[9] Vgl. IBM, 2013.
[10] Vgl. Lukoianova und V. L. Rubin, 2014.
[11] Vgl. Lukoianova und V. L. Rubin, 2014, S. 7.
[12] King, o.D.

das Anwenden von Algorithmen auf die Daten, wobei die Daten jedes Nutzers, mit denen aller anderen Nutzer verglichen werden.[13] Facebook hingegen nutzt die Möglichkeiten von Big Data unter anderem dazu, Vorschläge für neue Freunde zu machen oder dem Nutzer auf ihn zugeschnittene Werbung anzeigen zu können. Es gibt jedoch auch viele weitere Szenarien, in denen Big Data im Unternehmensumfeld zum Einsatz kommt. In der Medizin kann es eingesetzt werden, um individuelle Therapien für beispielsweise Krebspatienten, zu finden. Dies wird mit Hilfe von Algorithmen möglich, die Muster in der Diagnostik erkennen können. Im Finanzsektor wird es zur Fraud-Detection, also der Betrugserkennung, eingesetzt. Dabei werden Finanztransaktionen nach Unregelmäßigkeiten analysiert.[14] Darüber hinaus sind auch Echtzeitanwendungen besonders geeignet. Im Zusammenhang mit Internet of Things ist die Auswertung von Sensordaten in Echtzeit von besonderem Interesse.

Auch Netflix als großer Videostreaminganbieter setzt sehr stark auf die Big Data Technologie. Viele Entscheidungen bei Netflix werden datenbasiert getroffen. In dem Blogeintrag *Evolution of the Netflix Data Pipeline* erläutert das Real-Time Infrastructre Team von Netflix den Aufbau und die Leistungsfähigkeit ihrer Big-Data-Infrastruktur. Netflix nennt diese Infrastruktur Data Pipeline. Damit nimmt Netflix täglich rund 500 Milliarden Events (rund 1,3 Petabyte) auf, speichert und verarbeitet diese unter anderem in einer Hadoop-Umgebung. In Stoßzeiten werden hier pro Sekunde rund acht Millionen Events, mit insgesamt circa 24 Gigabyte Volumen aufgenommen.[15] Es gibt mehrere Hundert verschiedene Events, die auf diese Weise überwacht und analysiert werden. Mit Hilfe dieser Menge an Daten kann Netflix für ihr Geschäft wichtige Entscheidungen treffen. Dabei werden nicht nur triviale Events, wie welcher Inhalt wurde wie oft und von wem geschaut, sondern es werden weitaus tiefergehende Daten erhoben. Darunter sind beispielsweise wann wird zurück- oder vorgespult, auf Pause gedrückt, zu welcher Zeit man einen Inhalt wiedergibt, von wo aus man schaut, welches Endgerät genutzt wird und so weiter. Netflix weiß auch, wann in einem Film der Abspann beginnt. Dies wird beispielsweise dazu genutzt, um zu analysieren, was Nutzer tun, wenn der Abspann beginnt. Beenden sie die Netflix-App oder suchen sie weitere Inhalte. Hier kommt auch der Netflix Algorithmus zum Tragen, der personalisierte Film- oder Serienvorschläge macht.[16] Interessant ist auch, dass Netflix die gesammelten Daten dazu nutzt, um

[13] Vgl. Ohlhorst, 2012, S. 11 f.
[14] Vgl. Plattner, 2013.
[15] Vgl. Wu u. a., 2016.
[16] Vgl Bulygo, 2013.

neue Serien zu entwickeln. Bestes Beispiel hierfür ist die Netflix-Erfolgsserie *House of Cards*. Bevor Netflix sich die Rechte an der amerikanischen Version der Serie sicherte, konnte Netflix anhand der gesammelten Daten eine tiefergehende Analyse durchführen. Durch die Daten wusste Netflix, dass die britische Version gute Einschaltquoten hatte und viele Nutzer den Film *The Social Network* vom Produzenten David Fincher von Anfang bis Ende schauten. David Fincher fungiert bei *House of Cards* ebenfalls als Produzent. Darüber hinaus wusste Netflix auch, dass Nutzer, die die britische Version schauten sich ebenfalls gerne Filme von Kevin Spacey und / oder David Fincher anschauten.[17] Kevin Spacey spielt bei *House of Cards* die Hauptrolle. Aufgrund dieser Erkenntnisse war Netflix normalen Fernsehsendern weit voraus, die ihre Entscheidungen für eine neue Serie in der Regel nicht auf Grundlage solcher Daten entscheiden, da sie solche überhaupt nicht besitzen. Im Artikel *Big Data Lessons from Netflix* bei Wired wird zusätzlich noch dargestellt, dass selbst das Cover der Serie nicht dem Zufall überlassen wurde. Dazu wurden Farbschema von anderen populären Serien analysiert und auf Basis dessen wurde das House of Cards Cover entwickelt.[18] Durch die Analyse der Cover können auch Fragen beantwortet werden wie „Welche Farben sprechen welche Kunden an?“ oder „Sollten für Netflix eigene Serien immer ähnliche Farben verwendet werden oder für verschiedene Zielgruppen unterschiedliche Farbschema?“. Diese Beispiele zeigen, wie weit Big Data im Unternehmen gehen und vor allem auch, inwieweit dies die Entscheidungsfindung beeinflussen kann.

2.3 Abgrenzung zum klassichen Data Warehouse

Die Abgrenzung des klassischen Data Warehouse zu Big Data wird schon durch die im vorangegangenen Abschnitt genannten Herausforderungen sehr klar. Klassische Data Warehouse-Systeme können der Schnelllebigkeit, den unterschiedlichen Datenarten und deren schierer Menge, nicht in dem Maße gerecht werden, wie es bestimmte Anwendungsfälle erfordern. Hasso Plattner beschreibt in seinem Beitrag zu Big Data, in der Enzyklopädie der Wirtschaftsinformatik, die Abgrenzung so, dass existierende Data Warehouse- oder Business-Intelligence-Systeme üblicherweise mit einer aufwändigen Aufbereitung der Daten arbeiten. Big Data Anwendungen hingegen können in der Regel ohne eine aufwändige Aufbereitung auskommen. Hierdurch lassen sich Kosten sparen, eine hohe Flexibilität und der

[17] Vgl. Bulygo, 2013.
[18] Vgl. Simon, 2014.

schnelle wie einfache Zugriff auf Analysen aktuellster Daten erreichen.[19]

2.4 Datenspeicherung

Um den An- und Herausforderungen (Abschnitt 2.1) von Big Data Anwendungen gerecht zu werden, reichen meist relationale Datenbanksysteme nicht aus, aufgrund dessen werden in diesem Abschnitt alternative Möglichkeiten der Datenspeicherung betrachtet. Zuerst werden die sogenannten NoSQL-Datenbanken (Not only SQL) näher beschrieben und die verschiedenen zugrundeliegenden Konzepte, sowie einige Vertreter, vorgestellt. Im Anschluss an diesen Abschnitt wird das Hadoop-System und seine Bestandteile erläutert. Dieses System spielt in vielen Big-Data-Lösungen eine tragende Rolle.

2.4.1 NoSQL-Datenbanken

Je nachdem, wo man in der Literatur recherchiert, wurden die ersten NoSQL-Systeme schon Ende der 1970er, Anfang der 1980er Jahre entwickelt.[20] Nach *Edlich u. a.*, gilt als erstes System die Key-Hash-Datenbank *DBM* von Ken Thompson. In den 80er Jahren wurden weitere NoSQL-Systeme wie Lotus Notes oder BerkeleyDB entwickelt. Der Begriff NoSQL tauchte allerdings erst 1998 auf, bei einer relationalen Datenbank ohne SQL-API.[21] Bei NoSQL spricht man nicht von einer einzigen bestimmten Technologie, sondern es handelt sich dabei eher um Konzepte und Ideen die von verschiedenen Gruppen oder Firmen auf ihre speziellen Rahmenbedingungen angewandt werden. Dadurch sind vor allem so viele unterschiedliche NoSQL-Datenbanken entstanden über die Jahre. Der eigentliche Erfolg und Fortschritt für NoSQL-Datenbanksysteme kam jedoch erst um die Jahrtausendwende auf, als die ersten Firmen, im Zuge des Web 2.0, versuchten große Datenmengen zu verarbeiten. Gerade Google ist hier mit seinen Entwicklungen, wie einem eigenen Filesystem (Google File System) und dem darauf aufbauenden Map/Reduce und BigTable-Datenbanksystem, eine Art NoSQL-Pionierrolle zuzuschreiben.[22] In den Jahren 2006 bis 2009 entstanden jedoch die klassichen NoSQL-Datenbanksysteme, wie unter anderem MongoDB, Cassandra CouchDB. Der Begriff NoSQL wurde dann 2009 erneut verwendet, als Eric Evans von der Firma Rackspace und Johan Oskarsson von Last.fm zum NoSQL Meetup einluden. Sie brauchten einen Namen

[19] Vgl. Plattner, 2013.
[20] Vgl. Edlich u. a., 2010, S. 1.
[21] Vgl. Edlich u. a., 2010, S. 1; Vgl. Fowler, 2015, S. 8.
[22] Vgl. Edlich u. a., 2010, S. 1; Vgl. Fowler, 2015, S. 8.

für das Treffen und ebenfalls einen einfach zu teilenden Hashtag und entschieden sich daher für #NoSQL.[23] Es war das erste Mal, dass Menschen zusammen kamen, um dabei über nichtrelationale Datenbanken zu sprechen, die sie anschließend als NoSQL-Datenbanken bezeichneten.

2.4.1.1 Eigenschaften

Natürlich zeichnen sich solche NoSQL-Systeme durch bestimmte Eigenschaften aus, die sie auch gleichzeitig von klassischen relationalen System abgrenzen. Eine genaue Definition für NoSQL-Systeme ist jedoch recht schwierig, da es vor allem in der Anfangszeit weder Gremien noch Organisationen gab. Dies hat jedoch auch zur Vielfalt des NoSQL-Ökosystems beigetragen.[24] Je nach Literatur werden mehr oder weniger viele Eigenschaften genannt. Gemein haben alle Autoren jedoch die folgenden Eigenschaften:[25]

- Die Datenbank ist nicht relational
- Die Datenbank benötigt kein Schema beziehungsweise nur minimale Schemarestriktionen
- NoSQL-Systeme sind auf verteilte Architekturen ausgelegt

Die erste Eigenschaft ist, dass die Datenbank nicht relational ist. Eine relationale Datenbank speichert beispielsweise die Bestelldaten einer Bestellung in einer Tabelle der Datenbank und diese kann durch eine Relation mit einer weiteren Tabelle, den Kundendetails, verbunden sein. Bei NoSQL-Datenbanken, die nicht relational aufgebaut sind, existieren diese Beziehungen zwischen Tabellen nicht. Das bedeutet, dass die Bestelldaten mit den Kundendetails zusammen, in einem Tupel, abgebildet werden. Tupel wird meist synonym zu Datensatz verwendet. Ein Tupel kann aus mehreren Datenfeldern bestehen, den sogenannten Attributen. In relationalen Systemen wird darauf geachtet, dass die Daten möglichst normalisiert sind, bei NoSQL liegen die Daten in der Regel denormalisiert ab. Bei der Normalisierung versucht man, durch das Aufteilen von Attributen in eigenständige Tabellen (Relationen), Redundanzen weitestgehend zu minimieren. Der Vorteil einer denormalisierten Art der Datenspeicherung ist, dass bei einer Datenbankabfrage nicht zuerst die Informationen aus verschiedenen Tabellen zusammengefügt werden müssen. Dies ist immer auch mit einem höheren Rechenaufwand verbunden.

[23] Vgl. Fowler, 2015, S. 10.
[24] Vgl. Edlich u. a., 2010, S. 2.
[25] Vgl. Andreas Meier, 2016, S. 222; Vgl. Fowler, 2015, S. 12.

Die zweite Eigenschaft der NoSQL-Systeme liegt darin schemafrei oder nur mit sehr minimalen Schemarestriktionen funktionieren zu können. In dieser Eigenschaft liegt einer der größten Unterschiede zwischen klassischen relationalen und den NoSQL-Systemen. So ist bei relationalen Datenbanksystemen meist ein hoher Aufwand für das Design der Datenbank und der Erstellung eines Schemas nötig. Auch eine Erweiterung des Schemas ist dabei häufig sehr zeitintensiv und kann eine Datenbank für eine bestimmte Zeit lahmlegen. Durch das Web 2.0 wurden die Ansprüche an Webauftritte und Anwendungen im Internet immer höher.[26] Es muss agil und schnell auf Anforderungen reagiert werden sonst wird man als Unternehmen von der Konkurrenz abgehängt. Eine Schemaanpassung würde hier mit relationalen Systemen zu viel Aufwand bedeuten und könnte den Auftritt für eine bestimmte Zeit außer Kraft setzen. Für Kunden müssen gerade heute Webanwendungen immer erreichbar sein, einen Ausfall kann sich hier kein Unternehmen leisten. Hier spielen NoSQL-Systeme durch ihre Schemafreiheit ihren größten Vorteil aus. Dadurch, dass sie schemafrei arbeiten, muss sich der Entwickler zuerst einmal keine großen Gedanken machen, wie die Datenbank die Daten speichert und wie sie intern funktioniert.[27] Die Entwicklung kann schneller starten und es muss nicht erst großer Aufwand in das Datenbankdesign gesteckt werden. Daten können in beliebigen Strukturen abgelegt werden und man versucht die Verantwortung für das Schema in die Anwendung auszulagern.[28] In der Anwendung ist es meistens einfacher eine Anpassung vorzunehmen. *Edlich u. a.* beschreiben diesen Ansatz so: „So kann die Anwendung von Anfang an erweiterte Daten (z.B. mit einem Feld mehr) schreiben und ein Hintergrundprozess die Daten konvertieren und als neue Version schreiben,“[29]. Durch diese Art der Anpassung gibt es nur tolerierbare kurze Abstände, in denen alte Daten beim Lesen ausgegeben werden. Natürlich muss eine vorübergehende Inkonsistenz für die Anwendung tolerierbar sein. *Fowler* nennt eine alternative Interpretation der Schemafreiheit in seinem Buch *NoSQL For Dummies* „Schema on read“. Damit ist gemeint, dass beim Schreiben das Schema, in dem Daten gespeichert werden, nicht relevant ist. Lediglich wenn die Daten ausgelesen werden, muss das Schema, wie Daten abgelegt sind, bekannt sein. Diesen Teil übernimmt dann meist die Anwendung.

Die dritte Eigenschaft bezieht sich auf die Skalierbarkeit von NoSQL-Systemen.

[26] Vgl. Edlich u. a., 2010, S. 3.
[27] Vgl. Fowler, 2015, S. 12.
[28] Vgl. Edlich u. a., 2010, S. 3.
[29] Edlich u. a., 2010, S. 3.

Viele Webdienste generieren so viele Daten, sodass diese nicht in einer Datenbank auf einem einzelnen Server gespeichert und / oder verarbeitet werden können. Bei NoSQL-Datenbanken hat man sich dafür entschieden, dass man die Datenbank verteilt betreiben kann, dies bedeutet auf mehreren Servern und gegebenenfalls auch geographisch verteilt. Der Vorteil bei verteilten Systemen ist unter anderem, dass auf vergleichsweise günstigere Hardware zurückgegriffen werden kann, sogenannte *commodity server*.[30] Dadurch werden die Kosten bei der Hardwareanschaffung reduziert und trotzdem kann mit viel Rechenkapazität gearbeitet werden. Zusätzlich zur Verteilung der Datenbank ist auch eine horizontale Skalierbarkeit dieser ein sehr entscheidender Punkt. Unter horizontaler Skalierbarkeit, auch *Scale-out* genannt, versteht man, das Einfügen oder Entfernen von Knoten. Diese können dynamisch hinzugefügt werden und tragen dann einen Teil der Daten, beziehungsweise der Last, auf dem Rechnerverbund (Cluster).[31] Im Gegensatz dazu wird bei dem sogenannten *Scale-up* die Leistungskapazität eines Servers aufgestockt. Mit Hilfe der Verteilung der Daten können nicht nur Lasten besser verteilt werden, sondern es ermöglicht auch eine einfachere Realisierung von hoher Verfügbarkeit des Systems. Das wird dadurch möglich, dass die Daten ein- oder zweimal über das Cluster repliziert werden. Fällt ein Server aus, können alle Daten weiterhin genutzt werden, da sie auf einem anderen Rechnerknoten (Node) noch vorhanden sind.

Eine weitere oft genannte Eigenschaft von NoSQL-Systemen ist, dass sie Open Source sind. Open Source bedeutet, dass der Quellcode der Software von jedem eingesehen werden kann und die Software auch frei verfügbar ist. Diese Eigenschaft trifft auf viele Systeme aus dem NoSQL-Bereich zu, jedoch nicht auf alle. Systeme, wie beispielsweise Amazon SimpleDB, werden auch zu NoSQL-Systemen gezählt, auch wenn sie nicht Open Source sind.[32] Trotz der Tatsache, dass viele Systeme Open Source sind, stehen meist Unternehmen hinter diesen. Unternehmen von Datenbanken wie neo4j oder MongoDB entwickelten andere Geschäftsmodelle um mit ihren Open Source-Systemen Geld zu verdienen, unter anderem durch Enterprise-Versionen mit Support für Unternehmen.

Neben der Open Source-Eigenschaft findet man in der Literatur auch die

[30] Vgl. Fowler, 2015, S. 16.
[31] Vgl. Mohanty, Jagadeesh und Srivatsa, 2013, S. 75 f.; Vgl. Edlich u. a., 2010, S. 3.
[32] Vgl. Edlich u. a., 2010, S. 3.

Eigenschaft, dass die NoSQL-Datenbank eine einfache Programmierschnittstelle (API) haben muss.[33] Relationale Datenbanken nutzen meist SQL als Schnittstelle. Es ist zwar sehr mächtig und recht klar strukturiert, kann aber schnell unperformant und fehleranfällig werden. Aus diesem Grund legt man bei NoSQL Wert darauf, dass man eine möglichst einfache Schnittstelle besitzt. Bei CouchDB beispielsweise werden Datenbankoperationen mit Hilfe von REST-Anfragen umgesetzt.[34] Bei REST handelt es sich um eine Programmierschnittstelle, die auf HTTP-Anfragen basiert. Diese wird vorrangig für die Maschine-zu-Maschine-Kommunikation und meist bei Webservices eingesetzt. Auf der einen Seite werden durch einfache APIs auch neue Einsatzmöglichkeiten vorstellbar. Oft besitzen die APIs auf der anderen Seite jedoch eingeschränkte Funktionalität, wodurch sie weniger mächtig sind als SQL.

Auch im Punkt Konsistenzmodell unterscheiden sich klassische relationale Systeme von den NoSQL-Systemen. Relationale Systeme sind dafür bekannt sehr strikte Konsistenz- und Transaktionsforderungen zu besitzen.[35] Dies bedeutet, dass die Daten innerhalb des Datenbanksystems immer konsistent sein müssen und Inkonsistenzen nicht auftreten dürfen. Man spricht vor allem bei relationalen Datenbanksystemen von dem sogenannten ACID-Prinzip, dass sich wie folgt zusammensetzt:

- **A**tomicity
- **C**onsistency
- **I**solation
- **D**urability

Im Grunde besagt das ACID-Modell, dass, sobald Daten einmal geschrieben wurden, völlige Konsistenz vorliegt.[36] Dies ist vor allen Dingen bei kritischen Transaktionen sehr wichtig. Unter Transaktionen versteht man „an Integritätsregeln gebundene Datenbankoperationen, die Datenbankzustände konsistenzerhaltend nachführen“.[37]

[33] Vgl. Andreas Meier, 2016, S. 222; Vgl. Edlich u. a., 2010, S. 2 f.
[34] Vgl. Edlich u. a., 2010, S. 4.
[35] Vgl. Edlich u. a., 2010, S. 5.
[36] Vgl. Fowler, 2015, S. 42.
[37] Vgl. Andreas Meier, 2016, S. 136.

Atomicity besagt dabei, dass eine Transaktion ganz oder gar nicht abgeschlossen wird. Hat man beispielsweise eine Transaktion, die aus vier hintereinander ausgeführten Befehlen besteht und es werden nur zwei der vier Befehle ausgeführt, wird die Transaktion abgebrochen und die vorangegangen Veränderungen durch die ersten zwei Befehle wieder rückgängig gemacht. Werden hingegen alle Befehle wie gewünscht ausgeführt, dann wird die Transaktion erfolgreich abgeschlossen.[38]

Consistency, als zweite Eigenschaft des ACID-Modells, besagt, dass eine Transaktion die Daten immer von einem zu einem anderen Konsistenzzustand führt. Das bedeutet, dass die Datenbank vor der Transaktion konsistent und auch danach wieder konsistent ist. Während die Transaktion durchgeführt wird, können Schritte der Transaktion dazu führen, dass einzelne Konsistenzbedingungen temporär verletzt werden. Am Ende der Transaktion wird die Konsistenz jedoch wiederhergestellt und stellt so die Widerspruchsfreiheit der Daten sicher.[39]

Isolation stellt sicher, dass mehrere gleichzeitig ausgeführte Transaktionen sich nicht gegenseitig beeinflussen. Dies bedeutet, dass Transaktionen isoliert voneinander ausgeführt werden. So gibt es keine Seiteneffekte auf einzelne Transaktionen, die nicht gewollt sind.[40] Ein Beispiel für Isolation wäre, wenn zwei Nutzer nach einem Eintrag suchen und wenn keiner gefunden wird soll der Datensatz hinzugefügt werden. Würden die beiden Transaktionen nicht voneinander isoliert sein, könnten in diesem Beispiel Dubletten entstehen. Beide Nutzer lesen zur selben Zeit die Datenbank aus und sehen, dass kein Eintrag vorhanden ist. Als Folge dessen legen beide einen Eintrag an. Am Ende der beiden Transaktionen gibt es den Eintrag zweimal, obwohl er nur einmal vorhanden sein sollte. Isolation hilft also dabei Transaktionen voneinander zu trennen, damit diese sich nicht gegenseitig beeinflussen. Dies gilt auch für das Lesen von Zwischenständen einer Transaktion. Eine Transaktion könnte den Zwischenstand einer anderen Transaktion lesen, wodurch falsche Daten ausgelesen werden können. Auch hier hilft Isolation solche Fehler zu vermeiden, indem der Datensatz für die Dauer der Transaktion gesperrt wird.

Durability, als letzte Eigenschaft des ACID-Prinzips, steht dafür, dass die

[38] Vgl. Harrison, 2015, S. 9; Vgl. Andreas Meier, 2016, S. 136.
[39] Vgl. Fowler, 2015, S. 43; Vgl. Andreas Meier, 2016, S. 136.
[40] Vgl. Andreas Meier, 2016, S. 137.

Datenbankzustände solange erhalten bleiben und ihre Gültigkeit besitzen, bis sie von Transaktionen wieder verändert werden[41]. Um diese Anforderung zu erfüllen, muss sichergestellt werden, dass die Daten auch nach einem Hard- oder Softwareausfall weiterhin vorhanden sind[42]. Ein Logfile, für die Transaktionen, kann dabei helfen, verlorene Transaktionen nach einem Systemausfall wieder reproduzieren zu können. Dadurch kann dann die Dauerhaftigkeit der Daten sichergestellt werden.

Das ACID-Prinzip wird immer sehr stark mit relationalen Datenbanken assoziiert. Jedoch gibt es auch viele Datenbanksysteme, die zu den NoSQL-Systemen zählen, die ebenfalls ACID erfüllen. Als prominentestes Beispiel wäre hier die Graphendatenbank Neo4j zu nennen.[43]

Doch gerade bei verteilten Systemen, die für webbasierte Anwendungen genutzt werden, gibt es Eigenschaften, die meist wichtiger sind als die Konsistenzgewährleistung des ACID-Prinzips. Betreibt man beispielsweise ein soziales Netzwerk, ist es wichtiger eine hohe Verfügbarkeit aufweisen zu können. Das Netzwerk muss auch dann verfügbar sein, wenn unter anderem ein Knoten im Rechnerverbund ausfällt. Auf der anderen Seite ist es in diesem Fall nicht kritisch, wenn nicht alle replizierten Knoten konsistent sind und zeitlich verzögert aktualisiert werden.[44]

Eric Brewer, ein amerikanischer Informatiker, formulierte zur Besonderheit bei verteilten Systemen, das sogenannte CAP-Theorem, basierend auf seiner Forschung zu verteilten Systemen an der University of California in Berkeley.[45] CAP steht dabei für:

- **Consistency:** Im Vergleich zum ACID-Prinzip, wo es **eine** konsistente Version der Daten gibt, spricht man bei verteilten Systemen von Konsistenz, wenn mehrere Clients die gleichen Daten von verschiedenen replizierten Knoten lesen und konsistente Ergebnisse erhalten.[46]
- **Availability:** Diese Eigenschaft beschreibt eine hohe Verfügbarkeit, damit soll sichergestellt werden, dass der Betrieb der Anwendung nicht unterbrochen wird. Außerdem ist es für viele Anwendungen wichtig,

[41] Vgl. Andreas Meier, 2016, S. 137.
[42] Vgl. Harrison, 2015, S. 10.
[43] Vgl. Edlich, 2016.
[44] Vgl. Fowler, 2015, S. 44; Vgl. Andreas Meier, 2016, S. 148.
[45] Vgl. Edlich u. a., 2010, S. 31.
[46] Vgl. McCreary und Kelly, 2013, S. 30; Vgl. Andreas Meier, 2016, S. 148.

dass die Antwortzeiten des Systems möglichst kurz sind. Das beste Beispiel sind hier Online-Shops, denn gerade im E-Commerce-Bereich will der Kunde nicht auf den Shop warten müssen, sonst kauft er bei der Konkurrenz.[47]

- **Partition tolerance:** Mit Partition tolerance ist eine Art Ausfallsicherheit gemeint. Dabei ist die Anforderung an das System, dass es in der Lage ist, selbst, bei einem Ausfall einzelner Knoten oder Verbindungen zwischen den Knoten, trotzdem die Anfrage des Clients beantworten zu können. Dies bedeutet, dass das System durch einen Ausfall einzelner Komponenten nicht komplett ausfällt.[48]

Diese Eigenschaften sind, je nach Anwendungsfall des verteilten Systems unterschiedlich wichtig. Die Erkenntnis, auf der das CAP-Theorem aufbaut, ist jedoch, dass ein verteiltes System immer nur zwei dieser drei Eigenschaften gleichzeitig erfüllen kann (siehe Abbildung 2.1).

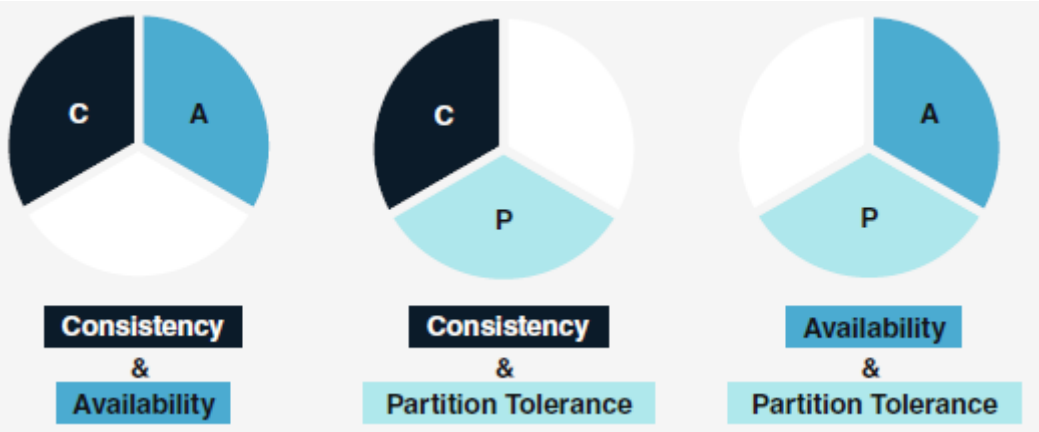

ABBILDUNG 2.1: Mögliche Ausprägungen des CAP-Theorems (Andreas Meier, 2016, S. 149)

Entsprechend dem CAP-Theorem ist es also immer notwendig zu überlegen, welche Kombination der drei Eigenschaften benötigt wird, für den einen konkreten Anwendungsfall. *Andreas Meier* stellt in seinem Buch *SQL- & NoSQL-Datenbanken* einige Beispiele für die verschiedenen Kombinationen dar. An der Börse beispielsweise, wird Wert auf Konsistenz und eine hohe Verfügbarkeit gelegt, demzufolge werden gerade dort Systeme die C und A des CAP-Theorems erfüllen, eingesetzt, meist handelt es sich dabei um relationale Systeme, die ähnlich dem ACID-Prinzip arbeiten.[49] Das System für Geldautomaten einer Bank müsste C und P erfüllen, da es konsistent

[47] Vgl. Edlich u. a., 2010, S. 32; Vgl. Andreas Meier, 2016, S. 148.
[48] Vgl. McCreary und Kelly, 2013, S. 31; Vgl. Edlich u. a., 2010, S. 32.
[49] Vgl. Andreas Meier, 2016, S. 149.

und ausfalltolerant sein muss. Als Beispiel für die Kombination aus Availability und Partition tolerance nennt *Andreas Meier* den Dienst Domain Name System (DNS), da dieser jederzeit verfügbar und ausfalltolerant sein muss um die Namen von Webseiten in IP-Adressen aufzulösen. Wäre dies nicht der Fall könnten Webadressen nicht aufgelöst werden.
Da durch die Forschungen Brewers und dem CAP-Theorem festgestellt wurde, dass man nicht alle drei Eigenschaften in verteilten Systemen zur gleichen Zeit umsetzen kann, wurde als Lösung des Konflikts das sogenannte BASE-Prinzip herangezogen.[50] BASE steht dabei für „Basically Available, Soft State, Eventually Consistent". BASE erlaubt es verteilten Systemen weichere Konsistenzanforderungen umzusetzen. Dabei können replizierte Knoten im Cluster temporär unterschiedliche Versionen ihrer Daten haben und diese werden, beziehungsweise können, verzögert aktualisiert werden. Im Vergleich zu ACID bedeutet das, dass man hier Konsistenz als Übergangsprozess definiert und nicht von einem fest definierten Zustand nach einer Transaktion. *Edlich u. a.* sprechen im Buch *NoSQL: Einstieg in die Welt nichrelationaler Web 2.0 Datenbanken* von einem optimistischen Konsistenzansatz, wohingegen ACID einen pessimistischen Ansatz verfolgt.[51] Man spricht auch von Eventually Consistency. Konsistenz wird irgendwann erreicht, dies ist nicht festgelegt und natürlich immer abhängig von Faktoren, wie beispielsweise der Anzahl zu replizierender Knoten. Es kann also nach einer Transaktion immer eine bestimmte Zeit geben, in der das System inkonsistent ist. Das Modell der Eventually Consistency kann dabei in verteilten Systemen unterschiedliche Ausprägungen haben. Werner Vogels, der Chief Technology Officer von Amazon, hat diese Ausprägungen in seinem Weblog zu verteilten Systemen definiert[52], welche sich auch in der Literatur[53] und wissenschaftlichen Papern[54] wiederfinden. Dabei gibt es fünf Ausprägungen:

- **Causal Consistency:** Prozesse sind kausal abhängig, wenn Prozess A einen Wert Y schreibt, Prozess B diesen Wert liest und danach einen Wert X in die Datenbank schreibt. Causal Consistency ist gegeben, wenn trotz der zeitlich nahe beieinander liegenden Prozesse A und B sichergestellt wird, dass auf allen zu replizierenden Knoten Prozess B den zuvor von A geschriebenen Wert Y erhält.

[50] Vgl. Edlich u. a., 2010, S. 33.
[51] Vgl. Edlich u. a., 2010, S. 33.
[52] Vgl. Vogels, 2007.
[53] Vgl. Celko, 2013, S. 12; Vgl. Edlich u. a., 2010, S. 34.
[54] Vgl. Terry u. a., 1994; Vgl. Perrin, Mostéfaoui und Jard, 2016.

- **Read-your-write Consistency:** Dabei handelt es sich um eine besondere Ausprägung der Causal Consistency, bei der ein Prozess, nachdem er einen Wert geändert hat, immer den aktualisierten Wert erhält und niemals eine ältere Version.
- **Session Consistency:** Hierbei wird dem Prozess während der Dauer einer Session, mit der er auf die Datenbank zugreift, die Read-your-write Consistency garantiert. Wird die Session beendet oder das System fällt aus, gilt die Garantie nur in einer neuen Session für die darin vorgenommenen Operationen.
- **Monotonic Read Consistency:** Ein Prozess bekommt bei einem Lesezugriff immer die aktuellste Version eines Wertes und niemals den einer älteren Version.
- **Monotonic Write Consistency:** Das System garantiert dem Prozess seine Schreiboperationen serialisiert auszuführen, um sicherzustellen, dass die Operationen in der Reihenfolge ausgeführt werden, wie sie angestoßen wurden.

Diese Ausprägungen können in Systemen auch kombiniert vorkommen. Laut *Celko* sind die Eigenschaften Monotonic Read und Monotonic Write die wünschenswertesten in einem Eventual Consistency System. Dadurch wird den Entwicklern von Anwendungen die Arbeit erleichtert, dem System wird eine weiche Konsistenz ermöglicht und trotzdem wird eine hohe Verfügbarkeit sichergestellt.[55]

Der Fokus bei ACID liegt auf der Konsistenz der Daten, wohingegen BASE den Schwerpunkt auf Verfügbarkeit legt.[56] In Abbildung 2.2 werden die Unterschiede zwischen ACID und BASE aufgezeigt.

[55] Vgl. Celko, 2013, S. 12.
[56] Vgl. Edlich u. a., 2010, S. 33.

ACID	BASE
Konsistenz hat oberste Priorität (strong consistency)	Konsistenz wird verzögert etabliert (weak consistency)
meistens pessimistische Synchronisationsverfahren mit Sperrprotokollen	meistens optimistische Synchronisationsverfahren mit Differenzierungsoptionen
Verfügbarkeit bei überschaubaren Datenmengen gewährleistet	hohe Verfügbarkeit resp. Ausfalltoleranz bei massiv verteilter Datenhaltung
einige Integritätsregeln sind im Datenbankschema gewährleistet (z. B. referenzielle Integrität)	einige Integritätsregeln sind im Datenbankschema gewährleistet (z. B. referenzielle Integrität)

ABBILDUNG 2.2: Vergleich zwischen ACID und BASE (Andreas Meier, 2016, S. 154)

In der Literatur findet sich jedoch auch der Hinweis, dass es nicht entweder BASE oder ACID sein muss. Meist wird von einem Spektrum gesprochen, bei dem der eine Pol ACID und der andere Pol BASE darstellt.[57] NoSQL-Systeme sind meist in der Nähe des BASE-Pols angesiedelt, wohingegen relationale Systeme in der Regel am ACID-Pol stehen.

2.4.1.2 Typen

NoSQL-Systeme werden allgemein meist durch die vorangegangenen Eigenschaften klassifiziert. Innerhalb der NoSQL-Systemfamilie gibt es jedoch unterschiedliche Typen für verschiedene Anwendungsfälle. Meist unterscheiden sich die Typen durch die zugrundeliegende Architektur und Arbeitsweise des Systems. Grundsätzlich unterscheidet man zwischen folgenden Haupttypen:

- **Key-Value Stores**
- **Wide Column Stores**
- **Document Stores**
- **Graphendatenbanken**

Bei Key-Value Stores handelt es sich um die einfachste Datenbankarchitektur der NoSQL-Systeme und diese zählen zu den ältesten Datenbanken, da sie schon seit den 70er Jahren Verwendung finden.[58] Doch im Zuge des Web

[57] Vgl. V., 2015; Vgl. Edlich u. a., 2010, S. 34.
[58] Vgl. Edlich u. a., 2010, S. 131.

2.0 haben solche Systeme einen großen Aufschwung erlebt, da viele der Daten, die im Internet heutzutage anfallen, nicht relational miteinander verbunden sind und sich gerade dann Key-Value Stores besonders eignen. Hinzu kommt, dass große Serviceanbieter ihre Datenbanken über viele tausende Rechnerinstanzen verteilen und die Skalierung mit relationalen Datenstruktur sehr schwierig ist. Deswegen ist gerade die Skalierbarkeit, durch die nicht relational miteinander verbundenen Daten, der große Vorteil der Key-Value Stores. Eine andere Eigenschaft, die diese Art der Datenbanksysteme auszeichnet, ist die Einfachheit des Systems an sich. Als Beispiel nennt *Fowler* in seinem Buch *NoSQL For Dummies* die Datenbank Redis, welche aus nur rund 20.000 Zeilen Code besteht.[59] Die Key-Value Datenbanken arbeiten schemafrei, es gibt also keine Restriktionen, wie die Schlüssel-Wert-Paare abgelegt werden, lediglich der Schlüssel muss eindeutig sein. Diese Einfachheit macht es Entwicklern leicht, Anwendungen auf einer solchen Datenbank aufzubauen, jedoch muss die Anwendung dann mehr Aufgaben im Handling der Daten übernehmen. *Wiese* beschreibt Key-Value Stores in ihrem Buch *Advanced Data Management: For SQL, NoSQL, Cloud and Distributed Databases* so: „The characteristic feature of a key-value store is that it is ‚simple but quick': Data are stored in a simple key-value structure and the key-value store is ignorant of the content of the value part,".[60] Diese Zusammenfassung beschreibt sehr gut was einen Key-Value Store ausmacht und worin seine Vorteile liegen. Die Einfachheit hat jedoch auch ihren Preis. Auf der einen Seite bieten Schlüssel-Wert-Datenbanksysteme eine hohe Skalierbarkeit, Einfachheit und Performance, auf der anderen Seite jedoch bietet das Datenbanksystem selber nur wenige Funktionen. In der Literatur findet man hier unterschiedliche Angaben. Nach *Celko* gibt es vier Basisoperationen.[61] Dies sind das Einfügen, Löschen, Aktualisieren und Finden von Paaren in der Datenbank. *Wiese* hingegen beschreibt nur drei Basisoperationen, die grundsätzlich zur Verfügung stehen, nämlich das Schreiben, Lesen und Löschen von Paaren.[62] Bei einem Key-Value-Paar muss der Value nicht zwingend ein String, sondern kann auch beispielsweise eine Liste oder ein Array sein. Im Vergleich zu konventionellen relationalen Datenbanksystemen ist der Funktionsumfang deutlich eingeschränkter, jedoch überwiegen bei den meisten Einsatzszenarien der Key-Value Stores die Eigenschaften Performance und Skalierbarkeit.

[59] Vgl. Fowler, 2015, S. 97.
[60] Wiese, 2015, S. 105.
[61] Vgl. Celko, 2013, S. 81.
[62] Vgl. Wiese, 2015, S. 105.

Bei den Wide Column Stores handelt es sich um spaltenorientierte Datenbanksysteme. Die Idee stammt aus den 80er Jahren, richtigen Aufschwung und das Entstehen der heutigen Architekturen erlebte die Idee jedoch mit Googles BigTable und Amazons Dynamo. Klassische relationale Datenbanksysteme arbeiten zeilenorientiert. Dies bedeutet, dass alle Felder (Attribute) eines Tupels (Tabellenzeile) hintereinander abgelegt werden.[63] Müssen komplette Datensätze gelesen werden ist dies die einfachste Variante. Die spaltenorientierte Datenbank unterscheidet sich dadurch, dass einzelne Spalten der Tabelle in einer eigenen Datei abgelegt werden können, was zur Folge hat, dass in Lese-Reihenfolge nicht das nächste Attribut des Tupels gelesen wird, sondern das gleiche Attribut des nächsten Tupels.[64] Durch diese Art der Speicherung können kostengünstig große Datenmengen persistent gespeichert werden.[65] Hilfreich ist die Art der Speicherung, auch wenn Werte in Spalten aggregiert werden müssen, da dafür nicht ganze Tupel gelesen werden müssen, sondern nur die jeweilige Spalte. Werden also bestimmte Werte von einzelnen Attributen gesucht, weisen spaltenorientierte Systeme eine hohe Performance auf. Müssen jedoch ganze Tupel ausgelesen werden, über viele Spalten hinweg, verlangsamt sich der Zugriff. Die Wide Column Stores unterscheiden sich jedoch dahingehend von den reinspaltenorientierten Systemen, dass sie zusätzlich mit zeilenorientierten Ansätzen kombiniert werden.[66] Dabei werden bestimmte zusammenhängende Spalten in sogenannten Column Families gruppiert. Dies hilft dabei die Performance bei Lese-/Schreibvorgängen zu verbessern, da Spalten, die häufig zusammengelesen werden, zusammen gruppiert werden.[67] In einem klassischen relationalen System wäre das Equivalent eine Tabelle. Jedoch kann eine Wide Column Store Tabelle n Column Families besitzen, die wiederum aus n Spalten bestehen können. Aus diesem Grunde nennt man solche Systeme Wide Column Stores. Die Wide Column Stores sind unter anderem aus Googles Bigtable hervorgegangen, welches Google in dem wissenschaftlichen Artikel „Bigtable: A Distributed Storage System for Structured Data“ im Jahr 2008 beschreibt.[68] Darin beschreibt Google seine Bigtable Architektur, die darauf ausgelegt ist Petabytes an Daten, über tausende Server verteilt, zu verwalten. Google selbst nutzt die Technologie unter anderem für seine Suchmaschine oder Dienste wie Google Earth. Bigtable kann

[63] Vgl. Bößwetter, 2009, S. 61.
[64] Vgl. Bößwetter, 2009, S. 62.
[65] Vgl. Meier, 2016, S. 421.
[66] Vgl. Meier, 2016, S. 421.
[67] Vgl. Andreas Meier, 2016, S. 226.
[68] Vgl. Chang u. a., 2008.

dabei verschiedene Anforderungen an das System bedienen, wie Chang schreibt „[...] both in terms of data size (from URLs to web pages to satellite imagery) and latency requirements (from backend bulk processing to real-time data serving)“.[69] Es kann also für verschiedene Zwecke eingesetzt werden und bedient dabei verschiedene Anwendungen von Google. Google beschreibt Bigtable als „sparse, distributed, persistent multidimensional sorted map“.[70] Diese sogenannte Map wird mit Hilfe eines Zeilenschlüssels, einem Spaltenschlüssel und einem Zeitstempel indiziert. Der Zeitstempel dient dazu mehrere Versionen eines Wertes in einem Feld hinterlegen zu können. So sind beispielsweise aus Googles Bigtable dann Wide Column Store Systeme, wie HBase oder Cassandra hervorgegangen.

Die Dokumentdatenbanken, oder auch Document Stores genannt, zählen ebenfalls zu den NoSQL-Systemen. Als Ursprung der Document Stores gilt Lotus Notes, das 1989 entwickelt wurde. Der große Unterschied zu den anderen Systemen ist, dass die Speichereinheit, in der Daten abgelegt werden, Dokumente sind. Dabei handelt es sich jedoch nicht um jedes beliebige Dokument, sondern man spricht bei Dokumenten von strukturierten Daten in Datensätzen.[71] Diese haben untereinander keine Beziehung. Dadurch entsteht eine strukturierte Sammlung von unterschiedlichen Daten.[72] Durch die einfache Sammlung von semi-strukturierten Daten ist eine Skalierung und die Replikation vergleichsweise einfach umzusetzen. Einer der großen Eigenschaften von Dokumentdatenbanken ist die Schemafreiheit. Die Verantwortung für das Schema wird komplett an die Anwendung abgegeben. Muss für die Anwendung eine bestimmte Normalform oder referenzielle Integrität sichergestellt werden, dann ist die Dokumentdatenbank im Nachteil.[73] Auf der anderen Seite gibt es agile Web-Anwendungen jedoch einige Vorteile, da sich die Anwendung selber um Schemaänderungen oder Inkompatibilitäten kümmern kann.

Von ihrer Funktionsweise arbeitet eine Dokumentdatenbank auf der obersten Ebene wie ein Key-Value Store, da zu einem Schlüssel, der sogenannten Dokument-ID, ein Datensatz als Wert gespeichert wird. Im Vergleich zum Key-Value Store ist der Datensatz bei der Dokumentdatenbank ein Dokument. Auf der Dokumentenebene haben die Dokumente eine eigene interne

[69] Chang u. a., 2008, S. 1.
[70] Chang u. a., 2008, S. 1.
[71] Vgl. Andreas Meier, 2016, S. 229.
[72] Vgl. Meier, 2016, S. 421.
[73] Vgl. Edlich u. a., 2010, S. 101.

Struktur.[74] Wie bereits erwähnt, handelt es sich bei den Dokumenten nicht um beliebige unstrukturierte Daten, sondern um eine Datei mit strukturierten Daten, beispielsweise im JSON-Format. Innerhalb dieser Dokumente existiert eine Struktur anhand einer Liste von Attribut-Wert-Paaren, die wiederum Listen von Attribut-Wert-Paaren enthalten können.[75] In Abbildung 2.3 ist ein Beispiel für den Aufbau eines Dokuments abgebildet. Zu Beginn steht immer der Schlüssel des Dokuments (_id) und anschließend die weiteren Attribut-Wert-Paare. Eine Besonderheit in diesem Beispiel ist der Revisionsschlüssel (_rev), der für die Versionisierung der Dokumente genutzt wird. Am Attribut visitHistory sieht man, dass Attribut-Wert-Paare wiederum eine Liste von Attribut-Wert-Paaren enthalten und so verschachtelt aufgebaut sein können.

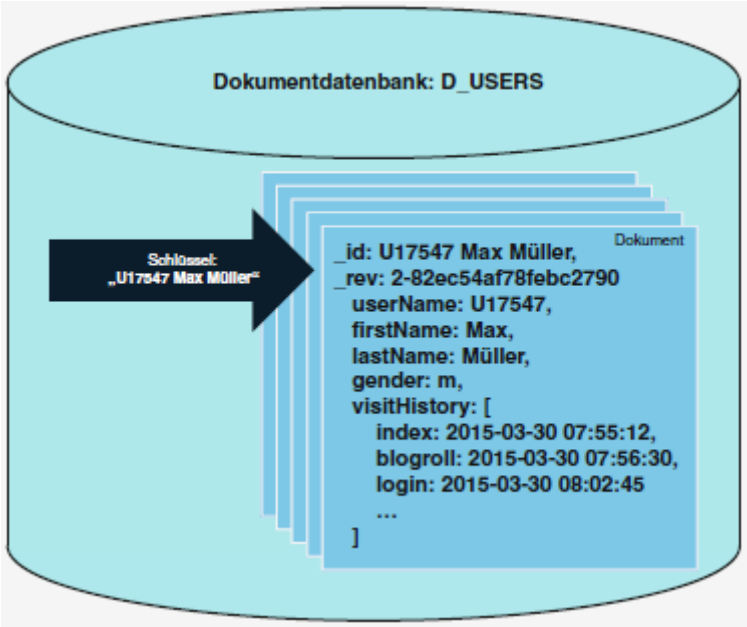

ABBILDUNG 2.3: Beispiel einer Dokumentdatenbank (Andreas Meier, 2016, S. 230)

Der letzte betrachtete Haupttyp der NoSQL-Systeme ist die Graphendatenbank. Bei den Graphendatenbanken handelt es sich um einen sehr speziellen Typ der NoSQL-Systeme. Die Idee hinter solchen Systemen ist nicht neu, sondern wird schon seit den letzten 30 Jahren erforscht.[76] Im Vergleich zu den zuvor genannten Systemtypen verzichten Graphendatenbanken nicht auf Schemata oder referenzielle Integrität. Dabei bedienen sich die Systeme des sogenannten Property Graphs (Eigenschaftsgraph). Daten werden in Knoten (Objekte) und gerichteten Kanten (Beziehungen) gespeichert. Der

[74] Vgl. Andreas Meier, 2016, S. 229.
[75] Vgl. Edlich u. a., 2010, S. 103.
[76] Vgl. Edlich u. a., 2010, S. 171.

Name Eigenschaftsgraph kommt daher, dass die Knoten und Kanten sowohl einen Namen haben, als auch Eigenschaften aufweisen können.[77] Ein Attribut-Wert-Paar stellt dabei eine Eigenschaft dar. Ein Beispiel für einen Knoten wäre ein Fußballspieler mit den Eigenschaften Name und Position, auf der er spielt. Ein weiterer Knoten könnte dann ein Fußballverein sein, mit der Eigenschaft Name. Eine Kante vom Spieler zum Verein könnte dann „spielt bei“ heißen. Eine Eigenschaft der Kante könnte ein Datumsattribut sein, dass angibt, seit wann der Spieler bei diesem Verein spielt. Durch diese Beziehungen zwischen den Knoten wird ein strukturierendes Schema impliziert. Das bedeutet, dass das Hinzufügen von neuen und unbekannten Knoten- oder Kantentypen unproblematisch ist, da das Datenbankmanagementsystem aus den vorhandenen Angaben implizit eine Schemaänderung durchführt und damit den entsprechenden Typ erstellt.[78] Das ist einer der großen Vorteile gegenüber relationalen Systemen, wo ähnliche Anwendungen nur mit hohem Aufwand umgesetzt werden können. Ein weiterer Vorteil ist die indexfreie Nachbarschaft, da das System alle direkten Nachbarn finden kann, ohne alle Kanten in der Datenbank berücksichtigen zu müssen.[79] Im Vergleich zu relationalen Systemen ist die Performance für Abfragen höher und gleichzeitig der Aufwand deutlich geringer. Grundsätzlich sind Graphendatenbanken für alle Anwendungen sinnvoll, die auf vernetzten Daten aufbauen. Beispiele für Anwendungsfälle sind Routenplanung für Navigationssysteme, Webseiten-Ranking bei Suchmaschinen oder Empfehlungssysteme in sozialen Netzwerken.[80] Der große Nachteil dadurch ist allerdings, dass das Sharding, also die Datenbankpartitionierung und Verteilung über mehrere Server, nicht funktioniert. Dies ist ein Resultat der Datenstruktur von Graphen. Ihren Vorteil spielen sie durch die Vernetzung der Knoten miteinander aus. Ein Aufspalten der Graphen in Teilgraphen ist nur schwer möglich.

2.4.2 Hadoop

Nach der Jahrtausendwende stieg das weltweite Datenaufkommen immer weiter an. Die Informatiker Doug Cutting und Mike Cafarella starteten das

[77] Vgl. Andreas Meier, 2016, S. 14.
[78] Vgl. Andreas Meier, 2016, S. 237.
[79] Vgl. Andreas Meier, 2016, S. 238.
[80] Vgl. Edlich u. a., 2010, S. 170.

Projekt Nutch, mit dem sie das Internet indizieren wollten, um eine Suchmaschine zu erstellen.[81] 2002 waren vor allem relationale Datenbanksysteme als Speicher genutzt worden. Diese waren jedoch einerseits teuer und andererseits schwer skalierbar. Dies änderte sich mit der Veröffentlichung von zwei wissenschaftlichen Artikeln durch Google, im Jahr 2003 und 2004, zum Google File System[82] und dem MapReduce-Algorithmus.[83] Google hatte damit Technologien entwickelt, die verteilt, skalierbar und performant waren. Diese Konzepte implementierten Cutting und Cafarella in ihrem Projekt und nannten diesen Teil Hadoop. Der Name stammt von Cuttings Sohn, der seinen Kuscheltierelefanten Hadoop nannte.[84] Hadoop ist ein Apache Open-Source Projekt. Apache beschreibt das Hadoop Framework wie folgt:

> „*The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.*“[85]

Diese kurze Erläuterung des Hadoop Frameworks von Apache selbst beschreibt Hadoop am besten. Auf der einen Seite erlaubt es das verteilte Verarbeiten von großen Datenmengen über große Computercluster, mit Hilfe einfacher Programmiermodelle. Dabei ist es auch kein Problem mehrere tausend Knoten zu verwenden, wodurch eine einfache Skalierung gegeben ist. Auf der anderen Seite bedarf es keiner Hardware für hohe Verfügbarkeit, da das Hadoop Framework Fehler und Ausfälle selbst bemerkt und diese auf Applikationsebene behebt. Es kann also sogenannte Commodity Hardware benutzt werden, die günstig und leicht austauschbar ist. Damit kann das Hadoop Framework günstig für hochverfügbare Dienste eingesetzt werden. Natürlich gibt es auch andere verteilte Systeme auf dem Markt, doch Hadoop sticht laut *Lam* durch folgende Eigenschaften hervor:[86]

[81] Vgl. Sitto und Presser, 2015, S. 1.
[82] Ghemawat, Gobioff und Leung, 2003.
[83] Dean und Ghemawat, 2004.
[84] Vgl. deRoos u. a., 2014, S. 13; Vgl. Sitto und Presser, 2015, S. 1.
[85] Apache, 2016.
[86] Vgl. Lam, 2011, S. 4 f.

- **Accessible:** Hiermit ist gemeint, dass Hadoop auf großen Clustern einfacher Hardware oder in Cloudumgebungen, wie Amazons Elastic Compute Cloud, betrieben werden kann.
- **Robust:** Da es auf Commodity Hardware läuft und man bei der Entwicklung von Hadoop den regelmäßigen Ausfall von Hardware berücksichtigt hat und es somit die meisten Fehler selbst behandeln kann.
- **Scalable:** Hadoop ist linear skalierbar durch einfaches Hinzufügen weiterer Knoten.
- **Simple:** Entwickler können schnell und einfach effizienten parallel ausführbaren Code schreiben.

Diese Eigenschaften machen Hadoop so erfolgreich. Es kann in kleinem Maßstab, in der Wissenschaft oder auch bei großen Unternehmen, wie Facebook, in großem Umfang eingesetzt werden. Grundlegend besteht Hadoop dabei aus lediglich zwei Bestandteilen:

- **Hadoop Distributed Filesystem** (HDFS)
- **MapReduce**

Dies ist jedoch immer abhängig von der Literatur und deren Erscheinungsjahr. Viele nennen nur diese beiden Komponenten als Hauptbestandteile von Hadoop.[87] Dies ist auf verschiedene Versionen des Hadoop Frameworks zurückzuführen. Mit Version 2.x wurde zusätzlich YARN (Yet Another Resource Negotiator) eingeführt. Durch YARN wurde das Ressourcen Management und das Job Lifecycle Management in eine eigene Komponente ausgelagert. *Sitto und Presser* beschreibt in seinem Buch *Field Guide to Hadoop* als dritte Komponente das Hadoop Ökosystem, wobei es sich um eine Sammlung von weiteren Tools und Komponenten handelt, die neben dem HDFS und MapReduce in einer Hadoop-Umgebung existieren und eingesetzt werden können. Grundsätzlich ist ein Hadoop-System aufgebaut wie in Abbildung 2.4.

[87] Holmes, 2012; Turkington, 2013.

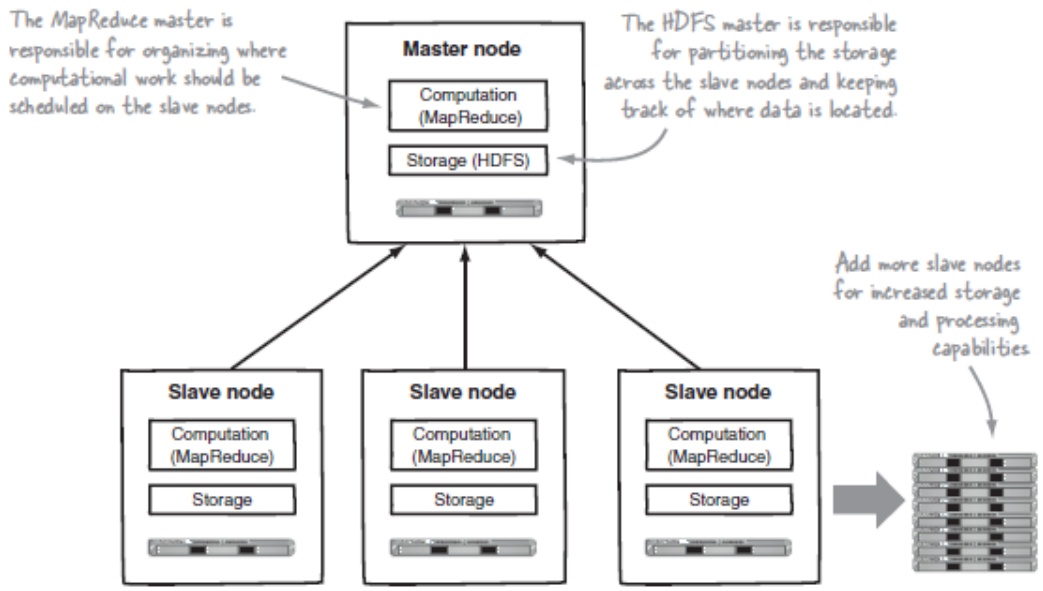

ABBILDUNG 2.4: Hadoop Architektur (Holmes, 2012, S. 5)

Dabei arbeitet das System nach dem Master-Slave-Prinzip. Der Master-Knoten ist gleich aufgebaut wie die Slave-Knoten. Dabei unterscheiden sich die Master für MapReduce und das HDFS dahingehend, dass diese jeweils für die Slaves verantwortlich sind. Das bedeutet, dass der MapReduce-Master dafür zuständig ist, wann und auf welchen Slaves Berechnungen angestoßen werden. Der HDFS-Master ist für die Partitionierung der Daten über die Slaves verantwortlich und überwacht, wo welche Daten abgelegt worden sind. In den folgenden Abschnitten werden die Komponenten HDFS, MapReduce, YARN und Teile des Ökosystems näher vorgestellt.

2.4.2.1 Hadoop Distributed Filesystem

Das Hadoop Distributed Filesystem ist die Speicherkomponente der Hadoop-Umgebung. Das Dateisystem basiert auf dem Google File System, das Google 2003 in einem wissenschaftlichen Artikel vorgestellt hat.[88] Google hatte mit GFS ein Filesystem entwickelt, das auf große Datenmengen zugeschnitten worden ist. Zum Zeitpunkt des Artikels im Jahr 2003 umfasste der größte Cluster mit GFS bei Google über 1000 Speicher-Knoten mit über 300 TB an Speicherplatz, welche von hunderten Nutzern kontinuierlich genutzt wurden.[89]

Dabei hat Google bestimmte Designentscheidungen für sein Filesystem getroffen. Unter anderem musste es sich selbst dauerhaft überwachen und

[88] s. Ghemawat, Gobioff und Leung, 2003.
[89] Vgl. Ghemawat, Gobioff und Leung, 2003, S. 2.

selbst Fehler entdecken und entsprechend darauf reagieren, da Ausfälle zum Alltag gehören. Da die meisten Dateien mehrere Gigabyte groß sind, sollten diese effizient verwaltet werden. HDFS ist auf einen hohen Durchsatz ausgelegt und kann damit sehr effizient große Dateien lesen. Die Performance ist bei vielen kleinen Anfragen nicht optimiert.[90] Bei der Entwicklung von GFS, respektive HDFS, wurde der Ansatz „write-once, read many“ verfolgt.[91] Dies bedeutet, dass das Dateisystem extra auf leseintensive Operationen ausgelegt wurde. Die dahinterstehende Annahme, die die Entwickler hatten, war, dass Dateien einmalig geschrieben und danach selten nochmal verändert werden.[92] Daher kommt der Ansatz „write once, read many“, weil die Dateien im Anschluss hauptsächlich gelesen werden. HDFS erlaubt es nicht eine bereits geschriebene Datei zu überschreiben. Anstelle dessen können nur Daten an die Datei über einen sogenannten „Append“ angehangen werden.[93] Das macht HDFS meist ungeeignet für sogenannte Online Transaction Processing Anwendungen (OLTP), bei denen Dateien häufig geändert werden müssen. Im Gegensatz dazu ist HDFS aber für das sequentielle Lesen von großen Dateien ausgelegt.[94]

Wie im vorherigen Abschnitt bereits kurz beschrieben und in Abbildung 2.4 abgebildet, arbeitet das Hadoop System mit einem Master-Slave-Cluster. Dies gilt entsprechend auch für das HDFS. Der Master-Knoten wird auch Name-Node genannt. Die Name-Node übernimmt alle Verwaltungsaufgaben und hält die Metadaten der auf den Slave-Knoten, den sogenannten Data-Nodes, gespeicherten Daten vor. Die Metadaten enthalten unter aanderem Informationen darüber, wann eine Datei erstellt wurde, wo die Blöcke einer Datei abgelegt wurden, wie viele Dateien auf einem Cluster liegen oder wie viele Data-Nodes im Cluster aktiv sind. Außerdem wird in den Metadaten hinterlegt, wo der Transaktionslog des Clusters zu finden ist. Damit wird sichergestellt, dass bei einem Ausfall bestimmte Operationen auf den Daten reproduziert werden können, um die Datenintegrität wiederherzustellen. Jeglicher Zugriff auf die Daten erfolgt immer über die Name-Node[95].

[90] Vgl. Turkington, 2013, S. 16.
[91] Vgl. Sitto und Presser, 2015, S. 3; Vgl. Turkington, 2013, S. 16.
[92] Vgl. Ghemawat, Gobioff und Leung, 2003, S. 2.
[93] Vgl. Sitto und Presser, 2015, S. 3.
[94] Vgl. Ghemawat, Gobioff und Leung, 2003, S. 2; Vgl. Sitto und Presser, 2015, S. 3.
[95] Vgl. Hurwitz, 2013, S. 113.

Ein großer Unterschied zu anderen Dateisystemen besteht in, der Blockgröße, in der HDFS Dateien gespeichert werden. Gängige Dateisysteme benutzten dabei eine Blockgröße von 4-32 KB. HDFS hingegen nutzt eine Blockgröße von 64 MB[96] oder 128 MB[97]. Diese Blöcke werden dann durch die Name-Node über das Hadoop Cluster verteilt. Dabei wird das HDFS auf den einzelnen Knoten nicht von dem jeweiligen Betriebssystem verwaltet, sondern HDFS bildet die Blöcke in Dateien auf dem vom Server zugrundeliegenden Dateisystem ab.[98] Um eine Ausfallsicherheit zu garantieren, bedient sich das HDFS der Replikation. Die Standardeinstellung dabei ist drei Replikate, die im Cluster verteilt werden. Eine Besonderheit bei der Replikation ist die sogenannte „Rack Awareness".[99] Da die Name-Node für jeden Cluster eine sogenannte Rack-ID in den Metadaten hinterlegt, spricht man von Rack-Awareness, weil die Name-Node dadurch unterscheiden kann, ob sich Data-Nodes des Clusters im gleichen oder in unterschiedlichen Racks befinden. Für die Replikation der Blöcke ist die Name-Node verantwortlich. Durch die Rack-ID können die Replikate auf jeweils eigene Racks verteilt werden. Dies ist jedoch nicht der optimale Weg, da es dadurch zu mehr Schreibvorgängen zwischen den Racks kommt, was wiederum den Netzwerkverkehr zwischen Racks erhöht. *Apache* und auch *Hurwitz* beschreiben die empfohlene Einstellung.[100] Das sogenannte *Block Placement* funktioniert dabei wie folgt: Die Name-Node platziert das erste Replikat auf einem anderen Server als dort, wo der Originalblock abgelegt wurde, jedoch im selben Rack. Damit bleibt ein Replikat des Blocks, im Falle eines Serverausfalls, im gleichen Rack. Für den Fall, dass ein komplettes Rack im Cluster ausfällt, platziert die Name-Node einen Block auf einem Server in einem anderen Rack. Damit man auch dort vor dem Ausfall des Servers geschützt ist ohne den Netzwerkverkehr zu erhöhen, wird ein weiteres Replikat auf einem anderen Server im zweiten Rack abgelegt. Der Netzwerkverkehr zwischen den Racks wird dadurch nicht erhöht, da HDFS hier auf sogenanntes *Replication Pipelining* setzt.[101] Eine Pipeline ist dabei eine Verbindung zwischen mehreren Data-Nodes. Vom Prinzip her wird nachdem ein Originalblock auf eine Data-Node geschrieben wurde, der Block als Replikat an die nächste Data-Node weitergeleitet. Diese wiederum leitet das Replikat an den nächsten von der Name-Node bestimmten Knoten weiter. Dies geht so

[96] Vgl. Turkington, 2013, S. 16.
[97] Vgl. White, 2016, S. 45.
[98] Vgl. Sitto und Presser, 2015, S. 3.
[99] Vgl. Apache, 2013; Vgl. Hurwitz, 2013, S. 114.
[100] Vgl. Apache, 2013; Vgl. Hurwitz, 2013, S. 116.
[101] Vgl. Apache, 2013.

weiter bis alle Replikate verteilt sind. Durch diesen Mechanismus muss der Block nur einmal zwischen verschiedenen Racks übertragen werden, die anderen Übertragungen geschehen rackintern. Greift ein Client auf eine Datei zu, versucht die Name-Node aus Performance-Gründen immer das Replikat, das sich am nächsten zum Client befindet, auszugeben.

Die Data-Nodes müssen regelmäßig sogenannte Heartbeatnachrichten an die Name-Node schicken, damit die Name-Node weiß, welche Data-Node aktiv ist. Fällt eine Data-Node aus oder die Netzwerkverbindung wird unterbrochen, registriert die Name-Node das Ausbleiben der Heartbeatnachricht dieses Knotens. In diesem Fall wird der Knoten durch die Name-Node als tot markiert und wird bei neuen Schreib- oder Leseanfragen nicht mehr berücksichtigt.[102] Die Blöcke, die auf dieser Data-Node gespeichert waren sind dann für das HDFS nicht mehr verfügbar und die Name-Node stößt den Prozess an, um wieder den Zustand von drei Replikaten neben dem Originalblock im Cluster herzustellen. In einem Cluster kann es vorkommen, dass bestimmte Data Nodes stark ausgelastet sind und andere Nodes leer sind, beispielsweise durch das Hinzufügen einer neuen Node. HDFS bietet für diesen Fall einen Rebalancer-Dienst. Dieser ist aktiv, wenn auch das Cluster aktiv ist und kann gegebenenfalls Daten zwischen den Data-Nodes verschieben, damit die Daten gleichmäßig verteilt sind und bestimmte Data-Nodes nicht volllaufen.[103]

Vor der Version 2.x waren die Name-Nodes im Cluster ein sogenannter „Single-Point-of-Failure“, da es keinen Mechanismus gab um eine Ersatzmaschine bei einem Ausfall zuzuschalten.[104] Fiel die Name-Node aus, musste manuell eingegriffen werden. Es gibt für die Name-Node einen Checkpoint-Prozess, dieser stellt das sogenannte *fsimage* her.[105] Daraus können bei einem Ausfall die Metadaten wiederhergestellt werden. Dieser Prozess kann unter anderem auf einem Name-Node ähnlichen Knoten durchgeführt werden, der *Secondary Name-Node* genannt wird.[106] Secondary ist dabei jedoch missverständlich, da es sich dabei nicht um einen Hot-Standby-Server, also eine Ersatz-Name-Node handelt. Die Name-Node pflegt ein Logfile das *edits* heißt, darin werden alle Änderungen an den Daten gespeichert. Wird ein bestimmtes Limit der edits-Datei erreicht, beginnt die Name-Node eine

[102] Vgl. Apache, 2013.
[103] Vgl. Hurwitz, 2013, S. 116.
[104] Vgl. Holmes, 2012, S. 13.
[105] Vgl. deRoos u. a., 2014, S. 64.
[106] Vgl. deRoos u. a., 2014, S. 64.

neue edits-Datei und übergibt die alte Datei und das fsimage an die Secondary Name-Node. Diese stellt dann, aus den Änderungen aus der edits-Datei und dem fsimage, ein neues fsimage her. Anschließend wird das fsimage wieder an die Name-Node übergeben und diese überschreibt das Alte mit dem neuen fsimage. Neben der Secondary Name-Node gibt es auch die Möglichkeit den Checkpoint-Prozess auf einer sogenannten Checkpoint Node oder einer Backup Node auszuführen. Diese werden in dieser Arbeit nicht näher betrachtet. Mit Version 2.x ist auch die Möglichkeit einer Standby-Name-Node eingeführt worden, so dass bei einem Ausfall auch ein Ersatzknoten einspringen kann.[107] Dafür teilen sich die aktive Name-Node und die Standby-Knoten einen hochverfügbaren Speicher, wo der Edit-Log abgelegt wird. Fällt der aktive Knoten aus, liest der Standby-Knoten den Edit-Log, aktualisiert seine Daten, anschließend übernimmt er die Aufgabe der ausgefallenen Name-Node. Die Metadaten der Blöcke hält die Standby-Node ebenfalls im Speicher vor und muss nur die Änderungen aus dem Edit-Log übernehmen, um konsistent zu sein. Diese Möglichkeit wurde HDFS High Availability genannt, da es dafür sorgt, dass der Cluster hochverfügbar ist und die Name-Node kein Single-Point-of-Failure mehr ist.

2.4.2.2 MapReduce

Bei MapReduce handelt es sich um ein Programmiermodell, welches große Datenmengen in einer verteilten Umgebung verarbeitet. Es ist neben HDFS einer der Hauptbestandteile des Hadoop-Systems. Genau wie HDFS basiert MapReduce auf einer Entwicklung Googles. MapReduce wurde von Google im Jahr 2004 im wissenschaftlichen Artikel „MapReduce“ vorgestellt. Der Name MapReduce kommt von den Funktionen *map* und *reduce*, die vor allem in funktionalen Programmiersprachen genutzt wurden.[108] Bisherige Ansätze von verteilten Berechnungen und Programmen stellten sich so dar, dass sich das Programm um die Hintergrundprozesse kümmern muss, wie beispielsweise das Verteilen der Daten und das Parallelisieren der Berechnungen. Bei MapReduce soll sich der Entwickler auf die Funktionen konzentrieren ohne sich mit den Hintergrundprozessen beschäftigen zu müssen. Der MapReduce-Prozess wird in drei Phasen unterteilt.

- **Map**
- **Shuffle and Sort**

[107] Vgl. White, 2016, S. 49.
[108] Vgl. Dean und Ghemawat, 2004.

- **Reduce**

Dabei arbeitet der MapReduce-Algorithmus immer mit Key-Value-Paaren.[109] Sowohl die Map- als auch die Reduce-Funktion werden durch den Benutzer geschrieben, der sich dabei der MapReduce-Library bedient.[110] Der Gedanke hinter der Funktionsweise der beiden Funktionen war, laut den Google-Ingenieuren, dass die meisten Berechnungen, die es zu lösen galt, immer nach einem ähnlichen Prinzip verliefen. Es sollte eine Map-Operation auf jeden Datensatz des Inputs angewendet werden und daraus Key-Value-Paare generiert werden. Werte, die den selben Key hatten, wurden mit einer Reduce-Operation auf, in der Regel, einen Wert reduziert. Grundlegend ist dies auch die Funktionsweise von MapReduce, nur dass die Berechnungen auf große Datenmengen, also Terabyte oder sogar Petabyte, umgesetzt werden können. Dabei werden die Berechnungen parallelisiert und die Teilberechnungen auf möglichst vielen Knoten gleichzeitig ausgeführt.

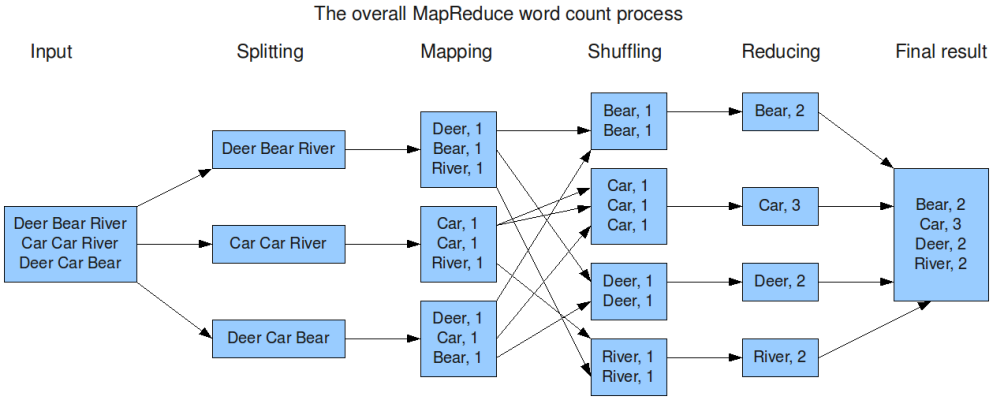

ABBILDUNG 2.5: MapReduce-Beispiel Wordcount (Quelle: http://bit.ly/1Sj4COu)

In Abbildung 2.5 ist in einem sehr vereinfachten Beispiel zu sehen, wie MapReduce funktioniert. Am Anfang des Prozesses stehen die Eingabedaten, in diesem Fall drei Zeilen mit je einem String aus drei einzelnen Wörtern. Das Beispiel zeigt einen MapReduce-Prozess um Wörter zu zählen. Wird das MapReduce-Programm vom Client gestartet, geht die Anfrage an den sogenannten JobTracker. Dieser Dienst läuft auf dem Master-Knoten. Der

[109] Vgl. Holmes, 2012, S. 7 f.; Vgl. Dean und Ghemawat, 2004, S. 2.
[110] Vgl. Dean und Ghemawat, 2004, S. 2.

JobTracker wurde in den Versionen vor 2.x genutzt und war dafür verantwortlich, alle Jobs zu verwalten.[111] Dies beinhaltet unter anderem das Parallelisieren der Berechnungen. Im Fall des Beispiels kann der JobTracker den Input auf drei Map-Instanzen aufteilen. Jeder Map-Instanz wird einer Zeile der Eingabedaten zugeteilt. Dabei nimmt der TaskTracker auf dem Slave-Knoten die Aufgabe entgegen, dieser ist für die Ausführung der Aufgabe zuständig. In der Map-Phase wird dann für jedes Wort ein Key-Value-Paar ausgegeben, wie beispielsweise „Bear, 1“. Das Zwischenergebnis ist eine Liste aus Key-Value-Paaren, wie sie in der Abbildung im Schritt Mapping zu sehen sind. Das Zwischenergebnis wird dann in der Shuffle-Sort-Phase zuerst nach dem Key sortiert, in diesem Fall ist das Wort der Schlüssel. Beim sogenannten Shuffling werden die Paare mit dem gleichen Schlüssel über ein Hashwertverfahren den Reducern zugeordnet.[112] So erhält jeder Reduce-Prozess eine bereits sortierte Liste. Die Reduce-Phase kann man sich so vorstellen, dass dort die Daten aggregiert werden. Im Fall des Beispiels werden die Werte der Wörter aufsummiert, um die Gesamtzahl eines Wortes in den Eingabedaten ausgeben zu können. Anschließend werden die Ergebnisse der Reducer beispielsweise in eine Datei im HDFS geschrieben.

Bei den Eingabedaten muss es sich natürlich nicht immer um Strings handeln. Konzeptionell müssen die Daten nur folgende Typen darstellen:[113]

$$\text{map } (k1, v1) \rightarrow list(k2, v2)$$
$$\text{reduce } (k2, list(v2)) \rightarrow list(v2)$$

Die Map-Instanzen bekommen ein Key-Value-Paar übergeben, aus dem sie eine Liste von Key-Value-Paaren generieren. Die Reduce-Instanz bekommt eine Liste von Werten zu einem Schlüssel nachdem Shuffle-Sort übergeben und generiert daraus eine Liste von Werten. Wie bereits beschrieben werden die Prozesse von dem Master-Knoten aus, durch den JobTracker, verwaltet. Dieser nimmt Client-Anfragen an und berechnet, wie viele Map- und Reduce-Instanzen benötigt werden und plant den Job entsprechend ein. Der JobTracker hat immer den aktuellen Status der Aufgaben, ob er *idle* ist, also unbearbeitet, *in-progess* oder *completed*. Für die Aufgaben, die nicht im Status idle sind, hält der JobTracker zusätzlich die Kennung des ausführenden Knotens.[114] Die TaskTracker auf den Slave-Knoten haben keine Verwaltungsaufgabe, sondern führen die Map- oder Reduce-Aufgaben aus, die sie

[111] Vgl. Prajapati, 2013, S. 31 f.
[112] Vgl. Sitto und Presser, 2015, S. 6; Vgl. Holmes, 2012, S. 7 f.
[113] Vgl. Dean und Ghemawat, 2004, S. 2.
[114] Vgl. Dean und Ghemawat, 2004, S. 4.

vom JobTracker zugewiesen bekommen.

Aufgrund der Tatsache, dass die MapReduce-Programme auch auf mehreren tausend Knoten ausgeführt werden können, wurde von den Ingenieuren bei Google auch ein Mechanismus für den Fall eines Ausfalls implementiert. Der JobTracker fragt regelmäßig bei den Slaves den Status der Berechnungen ab. Erhält er in einer gewissen Zeitspanne keine Antwort von diesem Knoten, markiert er diesen als *failed*.[115] Damit setzt der JobTracker alle Map-Aufgaben, die dieser Knoten bearbeitet hatte auf den Status *idle* zurück, da die Zwischenergebnisse der Map-Aufgaben lokal auf dem Knoten gespeichert werden. Zusätzlich werden alle Map- und Reduce-Aufgaben im Status *in progress* ebenfalls zurückgesetzt. Die zurückgesetzten Aufgaben können dann von anderen Knoten erneut ausgeführt werden. Fällt hingegen der JobTracker aus, kann dieser mit Hilfe des bereits beschriebenen Checkpoint-Prozesses wiederhergestellt werden. Ab Version 2.x von Hadoop wird das Ressourcenmanagement und damit auch die Job- und TaskTracker durch YARN ersetzt, welches im nächsten Abschnitt behandelt wird.

2.4.2.3 YARN

Mit Hadoop Version 2.x wurde YARN als weiterer Hauptbestandteil des Hadoop-Frameworks eingeführt. YARN steht dabei für „Yet Another Resource Negotiator“. Zuvor wurde die Verwaltung der Jobs durch MapReduce und die Job- und TaskTracker durchgeführt. Im Laufe der Zeit wurden die Datenmengen größer und die Anforderungen an ein Hadoop-Cluster immer größer und komplexer. Durch die Skalierung der Cluster auf mehrere tausend Knoten traten mit Versionen vor 2.x Engpässe auf. *Murthy u. a.* stellen in ihrem Buch *Apache Hadoop YARN* die Punkte heraus, die dazu geführt haben, YARN als Ressourcenmanagement zu entwickeln und zu implementieren. Nachdem Cluster extrem anwuchsen und immer mehr Jobs durch die Cluster bearbeitet wurden, kamen Probleme mit dem Speichermanagement auf.[116] Dies resultierte aus der Tatsache, dass die JobTracker Daten von den Jobs im Speicher halten. Wenn jedoch unlimitiert Daten durch beispielsweise einen Job, der unlimitiert einen Zähler hochzählt,

[115] Vgl. Dean und Ghemawat, 2004, S. 4.
[116] Vgl. Murthy u. a., 2014, S. 15.

entsteht, ist der TaskTracker nur noch mit dem Hochladen der Daten beschäftigt. Zudem ist die Speichernutzung des JobTrackers nicht effizient genug.[117] Auch die Ressourcennutzung auf den ausführenden Slave-Nodes ist in einer Hadoop-Umgebung vor Version 2.x nicht effizient gestaltet. Für Map- und Reduce-Jobs gibt es jeweils eine festgelegte Anzahl an Slots auf einer Node. Werden jedoch beispielsweise nicht alle Map-Slots genutzt, können diese nicht für Reduce-Aufgaben genutzt werden. Damit bleibt der Slot im schlechtesten Fall ungenutzt.

Neben der Speichernutzung war auch die Verlässlichkeit und Verfügbarkeit ein wichtiger Punkt, in dem MapReduce alleine nicht bestmöglich entwickelt wurde, da die JobTracker auf den Master-Knoten laufen und diese in einem Cluster einen Single-Point-of-Failure darstellen. Fällt der Knoten mit dem JobTracker aus, gehen damit alle laufenden Aufgaben im Cluster verloren und müssen nach der Wiederherstellung neu gestartet werden. Durch die sogenannte Downtime, also die Zeit wo der Master-Knoten nicht zur Verfügung steht, entsteht ein Rückstau an Jobs, die auf dem Cluster ausgeführt werden sollen. Bei der Wiederherstellung führt das wiederholte Ausführen der Jobs zu einer hohen Belastung auf dem Master-Knoten.

Ein weiterer Nachteil von MapReduce ist die Tatsache, dass man lediglich MapReduce-Jobs auf dem Hadoop-Cluster ausführen konnte. Für bestimmte Anwendungsfälle jedoch eignen sich andere Programmiermodelle besser, obwohl MapReduce für viele Fälle schon gut geeignet ist. Murthy u. a. nennen Machine Learning Algorithmen als Beispiel für Probleme, die sich besser mit anderen Programmiermodellen lösen lassen, da hierbei meist mehrere Iterationen nötig sind, um zum gewünschten Ergebnis zu kommen.[118] Natürlich könnte man dies auch mit mehreren MapReduce-Prozessen hintereinander lösen, der dadurch generierte Mehraufwand würde nur die Berechnungen verzögern. Daraus resultierte der Wunsch nach der sogenannten Programming Model Diversity, also der Möglichkeit, verschiedene Programmiermodelle auf dem Hadoop-Cluster anzuwenden.

[117] Vgl. Murthy u. a., 2014, S. 15.
[118] Vgl. Murthy u. a., 2014, S. 17.

Das war einer der Gründe für die Entwicklung von YARN. Vorher konnte auf dem Hadoop-System nur über MapReduce ein Job ausgeführt werden. Da MapReduce dabei auch die Verwaltung der Jobs übernommen hatte, konnten keine anderen Programmiermodelle Jobs auf einer Hadoop-Umgebung ausführen. Mit YARN wurde eine eigenständige Komponente für das Ressourcenmanagement geschaffen, die es auch anderen Programmen ermöglicht, in Hadoop ausgeführt zu werden. Dabei ist YARN jedoch abwärtskompatibel und es können ohne Probleme weiterhin MapReduce-Jobs ausgeführt werden. Dadurch ist MapReduce jedoch nur noch eins von mehreren möglichen Frameworks.[119]

Die Job- und TaskTracker, wie vor Version 2.x, existieren nicht mehr. Es wurden jedoch drei neue Dienste etabliert, der ResourceManager, der NodeManager und der sogenannte ApplicationMaster. Grundsätzlich wird bei YARN die Ressourcenverwaltung und das Überwachen der Jobs getrennt. Der ResourceManager befindet sich auf dem Master-Knoten und existiert nur einmal in einem Cluster. Dabei ist er für die Ressourcenverwaltung im Cluster verantwortlich. Entsprechend kann er die Ressourcen fair auf die eingehenden Jobs verteilen. Die Ressourcen werden mit Hilfe sogenannter Container verteilt. Dabei handelt es sich um ein logisches Paket aus Ressourcen, wie beispielsweise Speicher oder CPU-Kerne, die an einen bestimmten Knoten gebunden sind.[120] Um über den aktuellen Ressourcenzustand der Knoten im Cluster informiert zu sein, steht der ResourceManager in regelmäßigem Kontakt mit dem NodeManager, der sich auf jedem der Slave-Knoten befindet. Der NodeManager ist der auf den Knoten eingesetzte Dienst zur Überwachung der lokalen Ressourcen, dient dem Fehlerreporting und überwacht den Lebenszyklus, der auf dem Knoten laufenden Container. Aus den Berichten der NodeManger kann sich der ResourceManager einen Gesamtüberblick versschaffen. Der ApplicationMaster ist sozusagen der Master des Jobs den der Benutzer ausführen will und ist für alle Aspekte seines Lebenszykluses verantwortlich.[121] Dazu zählt auch dynamisch den Ressourcenverbrauch anzupassen oder den Ausführungsablauf zu verwalten. *Murthy u. a.* nennen hier als Beispiel einen MapReduce-Job, bei dem der ApplicationMaster dafür sorgt, dass die Reducer auf die Ausgabe der Map-Phase angewendet werden. Wie der Ablauf bei der Ausführung eines Jobs

[119] Vgl. Murthy u. a., 2014, S. 35.
[120] Vgl. Murthy u. a., 2014, S. 43.
[121] Vgl. Murthy u. a., 2014, S. 44.

auf einem Cluster mit Hilfe von YARN ist, wird in Abbildung 2.6 dargestellt.

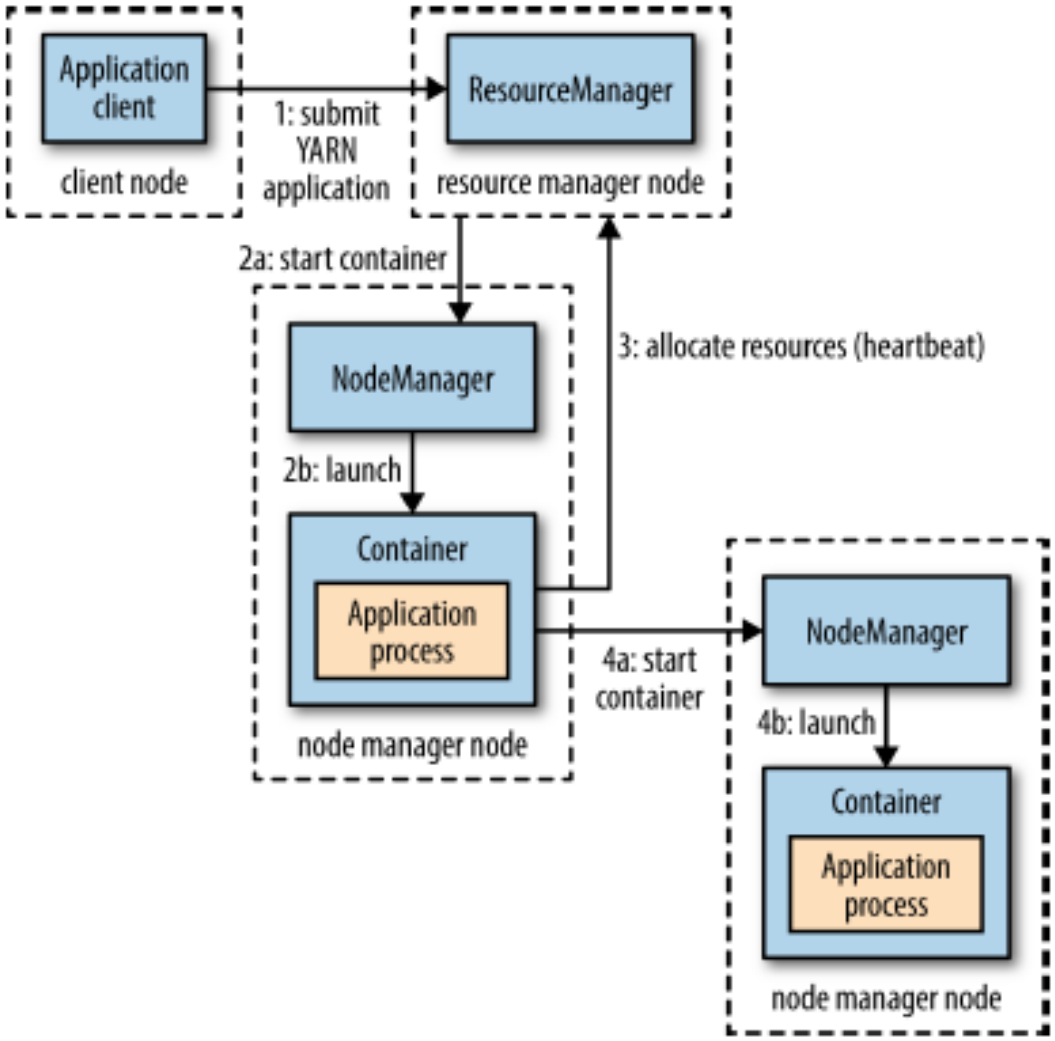

ABBILDUNG 2.6: YARN-Prozess (White, 2016, S. 80)

In Abbildung 2.6 wird im ersten Schritt eine Anfrage für das Ausführen einer Applikation von einem Client gestartet. Die Anfrage geht dabei an den ResourceManager des Clusters. Findet der ResourceManager einen Knoten, der genügend Kapazität hat um einen Container der Applikation auszuführen, weist er den NodeManager an, den Container auf diesem Knoten zu starten. Der wesentliche Unterschied zu den Versionen vor 2.x ist, dass der ResourceManager nur einen Container auf dem Knoten ausführen lässt. Es wird also nicht, wie beim JobTracker, erst berechnet wie der Job auf möglichst viele Knoten verteilt werden kann. Bei YARN ist dafür der ApplicationMaster verantwortlich. Der ResourceManager weist nur dem sogenannten „Container 0“ Ressourcen zu dessen Ausführung zu[122]. Benötigt der

[122]Vgl. Murthy u. a., 2014, S. 44.

ApplicationMaster zur Ausführung keine weiteren Ressourcen, dann wird die Anwendung in dem einen bestehenden Container ausgeführt. Benötigt der ApplicationMaster weitere Container, also Ressourcen, zur Ausführung der Applikation, fragt er diese beim ResourceManager an. Daraufhin kontaktiert der ApplicationMaster den NodeManager des vom ResourceManager zugewiesenen Knotens, um dort einen Container für die Applikation zu starten. Dadurch kann der ApplicationMaster dann eine verteilte Ausführung der Applikation realisieren. Bei einer Ressourcenanfrage können auch sogenannte „locality constraints“ angegeben werden[123]. Die Lokalität kann oft entscheidend für die Performance eines Clusters sein. Dies kann beispielsweise bedeuten, dass man einen weiteren Container anfragt, dieser sich aber im selben Rack befinden soll, damit der Netzwerkverkehr zwischen den Racks nicht unnötig erhöht wird und Daten schneller verschoben werden können, falls nötig[124].

2.4.2.4 Ökosystem

Die Hauptbestandteile des Hadoop-Systems sind eingebettet in ein ganzes Ökosystem. Zum Begriff Hadoop-Ökosystem finden sich im Internet sehr unterschiedliche Abbildungen, ein Beispiel für eine solche Abbildung ist in Abbildung 2.7 zu sehen. Die Abbildungen unterscheiden sich meist anhand der abgebildeten Komponenten um die Hauptkomponenten HDFS, YARN und MapReduce herum. Im Folgenden sollen einige Komponenten kurz vorgestellt werden, damit man diese im späteren Verlauf der Arbeit einordnen kann.

[123] Vgl. White, 2016, S. 81.
[124] Vgl. White, 2016, S. 81; Vgl. Murthy u. a., 2014, S. 44.

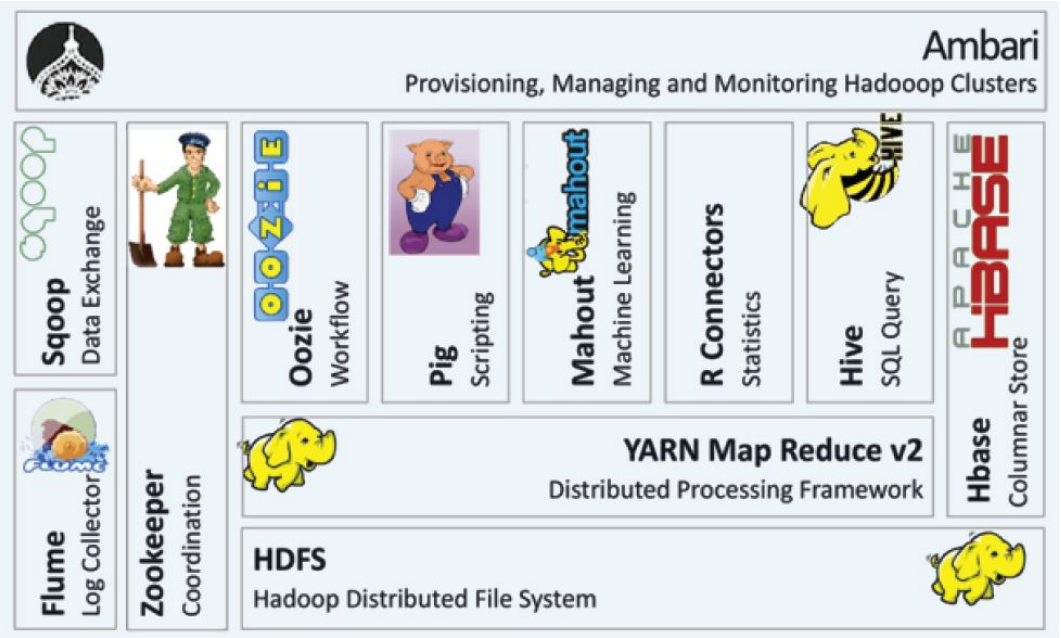

ABBILDUNG 2.7: Hadoop-Ökosystem (Quelle: http://bit.ly/2jQNNwN)

Mit HDFS bietet Hadoop ein verteiltes Dateisystem und MapReduce bietet die Möglichkeit verteilte Programme auszuführen. Es gibt jedoch keine Hauptkomponente, die dafür ausgelegt ist Daten effizient von externen Quellen in das Hadoop-System zu schreiben. Hier kommt Apaches **Flume** zum tragen. Dabei handelt es sich um ein Projekt der Apache Foundation, das sehr stark in Hadoop integriert ist. Apache selbst beschreibt Flume wie folgt:

> *„ Apache Flume is a distributed, reliable, and available system for efficiently collecting, aggregating and moving large amounts of log data from many different sources to a centralized data store.“*[125]

Das Zitat macht deutlich, wozu Flume entwickelt wurde. Es hilft dabei auf vergleichsweise einfache Art, effizient große Datenmengen von verschiedenen Datenquellen zu sammeln und an einem zentralen Speicherort abzulegen. Da Flume stark in Hadoop integriert ist, handelt es sich dabei in der Regel um ein Hadoop-System. Neben HDFS ist Flume jedoch auch in der Lage in andere Systeme zu schreiben, wie beispielsweise HBase.[126] Bei den Datenquellen, die dabei angeschlossen werden können, ist Flume nicht nur auf Log-Dateien beschränkt. Flume ist sehr anpassbar und kann daher Daten von den unterschiedlichsten Quellen beziehen. Darunter auch Daten aus Sozialen Netzwerken, wie Twitter oder Facebook und auch E-Mails.[127]

[125] Vgl. Foundation, 2012.
[126] Vgl. White, 2016, S. 381.
[127] Vgl. Foundation, 2012.

Flume ist für sogenannte Streamingdaten ausgelegt. Dies bedeutet, dass es speziell für die kontinuierliche Aufnahme von Daten entwickelt wurde. Gerade deswegen ist es besonders für die Aufnahme von Logs geeignet. Die Einheit in der Daten aufgezeichnet werden, nennt man auch Events, beispielsweise ein Eintrag im Log.[128]

Neben Flume existiert zur Speicherung von großen Datenmengen in Hadoop, mit **Hive** eine weitere häufig genutzte Komponente im Hadoop Ökosystem. Bei Hive handelt es sich um ein Data Warehouse-Framework, das auf Hadoop aufsetzt. Hive wurde von Facebook entwickelt, um einfach auf die großen Datenmengen zugreifen zu können, die Facebook täglich in ihrer Hadoop-Umgebung speichert.[129] Viele Datenanalysten können mit SQL (Structured Query Language), einer Datenbankabfragesprache, umgehen, meist jedoch nicht mit Sprachen, wie beispielsweise Java. In der Anfangszeit von Hadoop (Versionen vor 2.x) konnten nur MapReduce Jobs ausgeführt werden um die Daten in HDFS zu analysieren. Hive stellt mit HiveQL eine SQL-ähnliche Sprache zur Verfügung, mit der Daten im HDFS analysiert werden können. Dadurch sollte es Analysten mit SQL-Kenntnissen ermöglicht werden, große Datenmengen zu analysieren. Nach der Entwicklung bei Facebook wurde Hive ein Apache-Projekt und dort weiterentwickelt. Wird in Hive eine Abfrage mit HiveQL abgesetzt, so wird die Abfrage im Hintergrund von Hive in einen oder mehrere MapReduce-Tasks umgewandelt. Dadurch bedarf es keiner großen Lernphase einer neuen Programmiersprache, um die Daten in Hadoop analysieren zu können, sondern es können bereits vorhandene SQL-Kenntnisse genutzt werden.

Neben Hive existiert im Hadoop-Ökosystem auch eine Plattform namens **Pig**. Dabei handelt es sich, ähnlich wie Hive, ebenfalls um eine Komponente, die es ermöglicht, auf Daten in HDFS zuzugreifen und diese zu verarbeiten. Entwickelt wurde Pig bei Yahoo, um es den Forschern und Ingenieuren einfacher zu ermöglichen große Datenmengen zu analysieren.[130] Bei Pig werden Operationen mit Hilfe von der Sprache *Pig Latin* verfasst. Dabei handelt es sich um eine hohe Programmiersprache, die auf Datenflüsse ausgelegt ist. Es werden also Eingabedaten eingegeben und das Ergebnis der Operationen ausgegeben. Alle Operationen zusammen ergeben dann einen

[128] Vgl. deRoos u. a., 2014, S. 80.
[129] Vgl. Turkington, 2013, S. 238; Vgl. White, 2016, S. 471.
[130] Vgl. White, 2016, S. 423.

Datenfluss.[131] Der Name Pig kommt von „pigs eat everything“ und soll bedeuten, dass Pig mit unterschiedlichsten Formen von Eingabedaten arbeiten kann.[132] Der in Pig Latin geschriebene Code wird bei Ausführung, ähnlich wie bei Hive, in MapReduce Code umgewandelt und entsprechend auf dem Cluster ausgeführt. Die Antwort auf die Frage, ob man nun Hive oder Pig benutzen soll, hängt immer vom Anwendungsfall ab. Ist der Anwender SQL-erfahren und möchte das Ergebnis möglichst vorher definieren, ohne den Weg dahin festlegen zu müssen, sollte Hive genutzt werden, da Hive eher deklarativ ist. Pig hingegen ist imperativ und ermöglicht sehr genau, wie der Prozess ausgeführt wird.[133]

[131] Vgl. White, 2016, S. 423.
[132] Vgl. Sitto und Presser, 2015, S. 76.
[133] Vgl. Turkington, 2013, S. 354; Vgl. Warden, 2011, S. 13.

Kapitel 3

Data Mining

Das folgende Kapitel soll grob die Definition und die Anwendungsszenarien von Data Mining beschreiben.

3.1 Definition

In der Literatur finden sich unterschiedliche Definitionen zum Begriff Data Mining, viele unterscheiden sich vor allem in Details. Unter Data Mining versteht man allgemein eine Sammlung von rechnergestützten Prozessen und Methoden, mit denen es möglich ist, große Datenmengen zu analysieren. *Hand* definiert Data Mining als Analyse von meist großen Datenmengen, um unerwartete Beziehungen zu finden und diese für den Anwender nutzbar auszugeben. Diese Beziehungen, beziehungsweise Ergebnisse, werden meist als Muster oder Modelle bezeichnet[134].

Das Ziel von Data Mining ist also, Muster in einer Datenbasis zu erkennen. Dabei werden logische oder mathematische Beschreibungen genutzt, um diese Muster darzustellen, wie Decker und Focardi in ihrem wissenschaftlichen Paper von 1995 definieren:

> „*Data mining is a problem-solving methodology that finds a logical or mathematical description, eventually of a complex nature, of patterns and regularities in a set of data.*“[135]

Die Verfahren und damit die Analyse der Daten erfolgt bei Data-Mining-Methoden überwiegend selbstständig und unterscheidet sich daher vom klassischen Online Analytical Processing (OLAP), wo der Anwender viel selbst interagieren muss. Bodendorf beschreibt die Schritte, die Data-Mining-Verfahren ausführen, wie folgt:

[134] Vgl. Hand, 2001, S. 6.
[135] Decker und Focardi, 1995, S. 3.

- Die Verfahren können Hypothesen über mögliche Zusammenhänge, Muster oder Trends generieren,
- eine Hypothesenvalidierung anhand von Daten durchführen
- und aus den möglichen Hypothesen nur jene ausgeben, die als gültig anerkannt sind.[136]

Dabei handelt es sich natürlich um eine verallgemeinerte Darstellung des Vorgehens von Data-Mining-Verfahren. Die Hypothesengenerierung bei Data-Mining-Verfahren verfolgt dabei immer eine bestimmte Strategie, mit der nach Mustern in den Daten gesucht wird. Aufgrund der meist enorm großen Datenmengen wäre es nicht effizient, immer den gesamten Suchraum zu analysieren. Bodendorf beschreibt, dass die Verfahren Maßnahmen nutzen um Hypothesen gezielter aufzufinden.[137] Unter diesen Maßnahmen verbirgt sich beispielsweise das Nutzen von bereits gesammelten Informationen, beziehungsweise bereits vorher bewerteter Hypothesen. Dadurch können Teile des Suchraums ausgespart werden und die Generierung ist effizienter. Die Überprüfung der Hypothesen geschieht dann immer durch die Analyse der Basisdaten. Anhand dieser Analyse kann bewertet werden, ob eine Hypothese gültig ist oder als ungültig verworfen wird. Die Bewertung reicht von komplexen Maßen bis hin zu einfachen Ja- oder Nein-Entscheidungen.[138] Die aus der Bewertung resultierenden, für gültig bewerteten Hypothesen werden als Wissen ausgegeben und können auch zur Generierung neuer Hypothesen herangezogen werden.

Diese verallgemeinerte Darstellung von Data-Mining-Verfahren von Bodendorf spiegelt sich in ähnlicher Form auch in Bramers Ansicht, in seinem Buch *Principles of Data Mining*, wieder. Dabei beschreibt er Data Mining als Teil des Knowledge Discovery Prozesses, wie in Abbildung 3.1 zu sehen. Data Mining ist nur ein Teil eines größeren Prozesses. Data Mining hilft dabei Wissen zu generieren.[139] Die Abbildung 3.1 macht deutlich, dass gerade bevor das eigentliche Data Mining beginnt, schon ein komplexerer Prozess abläuft. Für gewöhnlich kommen die Daten, die analysiert werden, nicht nur aus einer einzigen Quelle, sondern aus mehreren unterschiedlichen. Diese werden zuerst einmal im Schritt *Integration* gebündelt, integriert und im Data Store dauerhaft gespeichert.

[136] Vgl. Bodendorf, 2006, S. 46.
[137] Vgl. Bodendorf, 2006, S. 46.
[138] Vgl. Bodendorf, 2006, S. 46.
[139] Vgl. Bramer, 2013, S. 2 f.

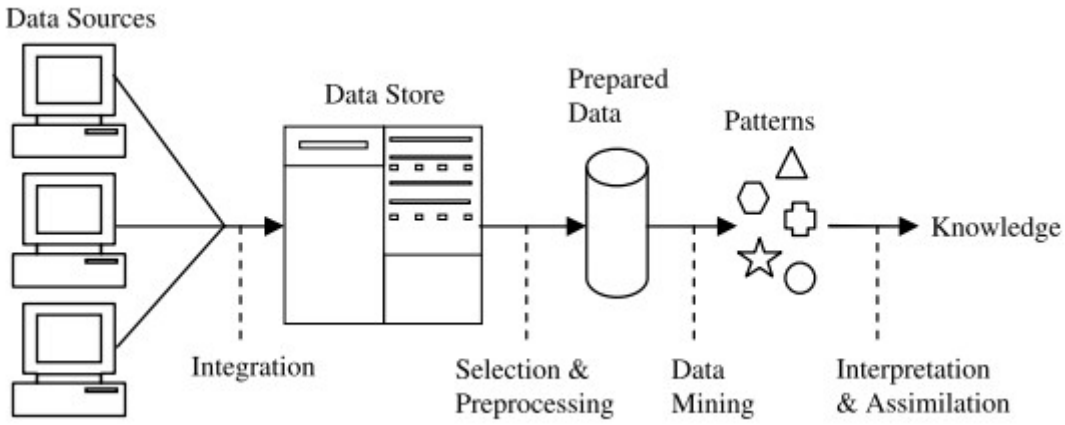

ABBILDUNG 3.1: Knowledge Discovery (Bramer, 2013, S. 2)

Häufig weisen die Daten aber noch nicht die Qualität auf, die man benötigt um sie analysieren zu können. Aus diesem Grund werden die Daten im Schritt *Selection & Preprocessing* bereinigt, also wie benötigt, weiterverarbeitet. Anschließend werden diese als vorbereitete Daten abgelegt. Auf Basis dieser Daten können dann die Data-Mining-Verfahren eingesetzt werden, um Muster und Zusammenhänge in den Daten zu entdecken. Die dadurch erstellte Ausgabe muss dann vom Anwender interpretiert und aufgenommen werden und es entsteht Wissen. Der *Knowledge Discovery* Prozess macht deutlich, wie wichtig nicht nur die Data-Mining-Verfahren als solche, sondern auch die Vorbereitung der Daten und auch die Ergebnisinterpretationen sind.[140]

3.2 Anwendungen

Im Data Mining gibt es verschiedene Anwendungsfälle und Herangehensweisen, um Daten zu analysieren. Die Wichtigsten werden in diesem Abschnitt kurz vorgestellt.

3.2.1 Klassifizierung

Bei der Klassifizierung handelt es sich um eines der gängigsten Verfahren. Dabei wird versucht, mit Hilfe eines Algorithmus, die Daten in Klassen zu unterteilen. Es wird also eine Vorhersage getroffen. Die Eingangsdaten sind keiner Klasse zugeordnet, weisen jedoch immer individuelle Merkmalkombinationen auf. Anhand dieser Kombinationen entscheidet der Algorithmus in welche Klasse ein Objekt eingeordnet wird. Ein Beispiel für einen solchen

[140] Vgl. Bramer, 2013, S. 3.

Algorithmus ist unter anderem das *Nearest Neighbour Matching*. Diese Methode betrachtet eine bestimmte Anzahl von *n* Nachbardatensätzen. Wenn beispielsweise eine Klassifizierung in Klasse A oder B vorgenommen wird und die angenommen 4 Nachbarn, entsprechend B, A, B und B, klassifiziert wurden, kann man annehmen, dass der nächste unklassifizierte Datensatz ebenfalls in die Kategorie B eingeordnet werden kann. Neben dieser Methode existieren jedoch noch weitere Algorithmen, mit denen eine Klassifizierung erfolgen kann.

3.2.2 Regressionsanalyse

Die Regressionsanalyse wird im Vergleich zu der Klassifizierung nicht dazu genutzt, um eine bestimmte Klasse vorherzusagen, sondern dient dazu, numerische Werte vorauszusagen. Klassische Beispiele sind für solche Anwendungen die Vorhersage von Unternehmensumsätzen oder Kursentwicklungen an der Börse.[141] Dabei wird versucht mit Hilfe der Regressionsanalyse einen funktionalen Zusammenhang zwischen den Variablen herzustellen. Der daraus resultierende mathematische Zusammenhang kann genutzt werden, um Vorhersagen für Werte zu treffen.

3.2.3 Assoziationsregeln

Eine weitere Methode des Data Minings sind die Assoziationsregeln. Häufig stellt sich die Frage, wie bestimmte Dinge miteinander in Beziehung stehen. Diese Beziehungen zwischen den einzelnen Variablen werden auch als Assoziationsregeln bezeichnet. Es ist immer wichtig, zu betrachten, wie verlässlich diese Regeln sind, da viele der Beziehungen keinen oder nur einen geringen Wert haben und damit nicht verlässlich sind.[142] Das gängigste und bekannteste Beispiel für Assoziationsregeln ist die Warenkorbanalyse. Die Intention dieser Analyse ist zu beobachten, welche Produkte häufig zusammen gekauft werden und damit in einer Beziehung stehen. Das Ergebnis der Analyse soll dazu beitragen, die Produkte im Laden besser zu platzieren. Das populärste Beispiel hierfür war eine Analyse der Supermarktkette Wal-Mart, die mit Hilfe von Data Mining und dem Assoziationsregel-Verfahren

[141] Vgl. Bramer, 2013, S. 7.
[142] Vgl. Bramer, 2013, S. 7.

einen Zusammenhang zwischen Bier und Windeln herstellen konnte. Wal-Mart fand heraus, dass Väter, die am Wochenende einkaufen gingen, sowohl Windeln für die Kinder, als auch Bier kauften. Daraufhin startete Wal-Mart den Versuch und positionierte die Bierpaletten in die Nähe der Windeln. Der Umsatz konnte durch diese Maßnahme weiter gesteigert werden.[143]

3.2.4 Clustering

Die letzte hier kurz vorgestellte Methode des Data Minings ist das Clustering. Das Clustering dient dazu Objekte, die sich ähnlich sind, in Gruppen zusammenzufassen. Ein einfaches Beispiel hierfür ist, dass ein Unternehmen seine Kunden nach bestimmten Kriterien, wie Alter, Umsatz oder ähnlichem gruppiert. Dies sind natürlich sehr einfache Beispiele. Bramer nennt in *Principles of Data Mining* auch komplexere Szenarien, beispielsweise den Versuch Länder mit ähnlicher Wirtschaft zu gruppieren oder in der Medizin Patienten mit ähnlichen Symptomen in einem Cluster zusammenzufassen.[144]

[143] Vgl. Rao, 1998.
[144] Vgl. Bramer, 2013, S. 311.

Kapitel 4

Sentiment-Analyse von Texten

Eine Besonderheit des Data Minings stellt das *Text Mining* dar. Dabei werden unstrukturierte Daten, in Form von Texten, analysiert. Das klassische Data Mining zielt vor allem auf strukturierte Daten ab, also Daten die zum Beispiel in geordneter Tabellenform vorliegen. Trotzdem können auf Texte auch die gleichen Verfahren angewendet werden, wie im klassischen Data Mining. Dazu müssen die Texte in ein numerisches Format transformiert werden. Dies kann beispielsweise die Präsenz eines Wortes innerhalb von Texten sein, wie in Tabelle 4.1 zu sehen.

Beruf	Personal	Weiterbildung	Karriere
1	1	0	1
1	0	1	0
0	1	1	0
1	1	1	0

TABELLE 4.1: Wortpräsenz in verschiedenen Dokumenten

Jede Zeile in der Tabelle stellt ein Dokument dar und jede Spalte ein Wort. Die Werte beschreiben die Anwesenheit (1), oder die Abwesenheit (0), des Wortes in diesem Dokument. Da die klassischen Data-Mining-Verfahren vor allem statistische Methoden sind, fehlt es ihnen an Wissen, bezüglich der Wortbedeutung und Grammatik.[145] Mit dieser Herangehensweise können also vor allem statistische Methoden auf Texte angewendet werden. Die üblichen Probleme, die mit dieser Art der Textrepräsentation gelöst werden können, sind die Klassifizierung und die Vorhersage. Dabei können beispielsweise Klassen für Dokumente vergeben werden, auf Basis der historischen und verfügbaren Daten. Durch die automatisierte Klassifizierung ist die Informationsgewinnung aus diesen Dokumenten einfacher, da ähnliche Dokumente schon in entsprechenden Klassen zusammengefasst wurden. So kann schneller auf ähnliche Dokumente zugegriffen werden. Neben diesen Verfahren gibt es noch viele weitere Methoden, Texte zu analysieren, dabei

[145] Vgl. Weiss, 2004, S. 4.

hilft unter anderem das Natural Language Processing (NLP). Dabei handelt es sich um das Verarbeiten natürlicher Sprache, wodurch dem Computersystem ermöglicht werden soll, die Bedeutung hinter Wörtern der menschlichen (natürlichen) Sprache und Zusammenhänge zwischen Wörtern zu verstehen. In diesem Kapitel wird speziell die Sentiment Analyse als Form der Text Analyse betrachtet. Dabei sollen auch das Natural Language Processing, sowie lexikonbasierte und Machine Learning Ansätze, betrachtet werden.

4.1 Sentiment Analyse

Die Sentiment Analyse ist auch häufig unter dem Begriff *Opinion Mining* oder *Sentiment Detection* zu finden. Im Zuge der Digitalisierung und der großen Verbreitung von sozialen Medien und Netzwerken bietet sich den Unternehmen eine große Datenquelle an Informationen über ihre Kunden. Der in den sozialen Netzwerken durch Benutzer generierte Inhalt bietet den Unternehmen die Möglichkeit, zu sehen, was der Kunde denkt, in dem er über Produkte schreibt oder seine Vorlieben preisgibt. Die Informationen, die auf der einen Seite aus diesen Daten generiert werden können sind enorm, auf der anderen Seite stellt die Art der Daten die Unternehmen vor Probleme. Nicht nur, dass Inhalte in sozialen Netzwerken in einer sehr hohen Geschwindigkeit generiert werden, sondern es handelt sich dabei um unstrukturierte Daten in Textform. Darüber hinaus werden die Texte meist sehr umgangssprachlich und mit Abkürzungen verfasst, was es für ein Computersystem noch einmal um einiges schwieriger macht, diese zu analysieren.[146]

Die Herkunft des Bedarfs nach Sentiment Analyse beschreibt *Ziegler* in seinem Artikel „Stummer Wächter“ so, dass dies aus den Anfängen der Reputation Intelligence hervorgegangen ist.[147] Bei Reputation Intelligence analysieren Unternehmen, beispielsweise wie häufig der Markenname oder bestimmte Produkte, in Quellen, wie Zeitungen, Blogs oder ähnlichem, genannt werden. Der Artikel stammt aus dem Jahre 2006, wo die Sentiment Analyse noch nicht weit fortgeschritten war und auch die Analyse-Methoden nicht so ausgeprägt waren. Für Unternehmen wurde es auch gerade durch

[146] Vgl. Markus Hofmann, 2016, S. 134 f.
[147] Vgl. Ziegler, 2006b, S. 119.

die Zunahme an benutzergenerierten Inhalten immer interessanter, die Meinungen der Nutzer zu analysieren. Dies bietet den Unternehmen die Möglichkeit, zu analysieren, ob bestimmte Produkte positiv oder negativ aufgenommen und in den Inhalten entsprechend dargestellt werden. In seinem anderen Artikel „Die Vermessung der Meinung“ beschreibt *Ziegler*, die Schwierigkeiten die mit dem Versuch subjektive Haltungen aus Texten zu extrahieren, einhergehen.

Als einer der größten Schwierigkeiten stellt Ziegler das Problem des Kontextes heraus. Betrachtet man Wörter wie „gut“ oder „schlecht“, ist die Polarität klar positiv, oder negativ. Als Beispiel, wo der Kontext für die Polarität des Wortes entscheidend ist, nennt *Ziegler* das Adjektiv „groß“.[148] Wird es im Zusammenhang mit einem Fernseher verwendet, kann davon ausgegangen werden, dass es im positiven Sinne eingesetzt wurde. Bei einem Produkt, das möglichst klein sein sollte, würde es eher negativ behaftet Verwendung finden.

Darüber hinaus erklärt *Ziegler*, dass eine sinnvolle Sentiment Analyse über Domänenwissen verfügen sollte. Damit ist beispielsweise bei einem Auto gemeint, dass dem Programm Informationen zu Unterthemen bekannt sind, wie Daten zum Motor, der Karosserie oder dem Design.[149] Entsprechende Lösungen sollen das, im zu analysierenden Text, genannte Subjekt erkennen und dann entsprechend die hinterlegten Informationen nutzen können.

Neben diesen beiden Punkten spielt auch die grammatikalische Analyse bei der Sentiment Analyse von Texten eine große Rolle. Dies ist im Hinblick auf die Sprachen für jede Sprache unterschiedlich zu betrachten, da jede Sprache ihre eigenen grammatikalischen Eigenheiten hat. Als Beispiel nennt *Ziegler* einen Konzessivsatz: „Obschon die Verarbeitung des T43 gut ist, reicht dieser nicht an das vergleichbare Modell von Fujitsu heran.“.[150] Anhand dieses Beispiels wird deutlich, wie schwierig es ist die Polarität eines Satzes zu bestimmen. Die erste Hälfte klingt positiv. Dieser wird jedoch nur dazu genutzt, um die negative zweite Satzhälfte zu verstärken. Im gleichen Zug nennt er die Valenzverschiebung durch Verneinung, Intensivierung oder modale Operatoren. wie beispielsweise „wenig begeistert“.[151]

[148] Vgl. Ziegler, 2006a, S. 106.
[149] Vgl. Ziegler, 2006a, S. 106.
[150] Vgl. Ziegler, 2006a, S. 107.
[151] Vgl. Ziegler, 2006a, S. 107.

Auch solche Konstellationen müssen bei der Analyse berücksichtigt werden, damit die Polarität eines Textes nicht verfälscht wird. Denn das Wort „begeistert“ ist klar positiv zu bewerten, wird jedoch durch das vorangehende Wort „wenig“ abgeschwächt.

Ziegler nennt in seinem Artikel von 2006 Ironie, Sarkasmus und Bildsprache als Grenzen der Sentiment Analyse, da es maschinell kaum umsetzbar wäre diese zu erkennen und entsprechend analysieren zu können. Heutzutage gibt es jedoch auch hier Ansätze, um entsprechenden Einsatz von Ironie und Sarkasmus erkennen zu können. Der wissenschaftliche Artikel „Identifikation und Analyse von ironischen und sarkastischen Kundenrezensionen im Web“ von *Schieber, Hilbert und Stillich* beschreibt wie ironische oder sarkastische Beiträge erkannt und entsprechend analysiert werden können. Dabei haben die Autoren des Artikels einen mehrstufigen Prozess zur Identifikation und Analyse von ironischen oder sarkastischen Texten entwickelt. Eine andere Gruppe von Wissenschaftlern hat bereits vor *Schieber, Hilbert und Stillich* ein Verfahren entwickelt um Sarkasmus zu identifizieren. In ihrem Artikel „ICWSM — A Great Catchy Name: Semi-Supervised Recognition of Sarcastic Sentences in Online Product Reviews“ stellen sie das Verfahren „Semi-supervised Algorithm for Sarcasm Identification“ vor.[152] Dabei bediente man sich eines halbüberwachten Klassifikationsansatzes, der auf Amazon-Bewertungen angewendet wurde und mit dem die Wissenschaftler eine Trefferquote von rund 83% bei der Erkennung von Sätzen mit Sarkasmus erreichten. Das zeigt deutlich, dass Ironie und Sarkasmus nicht zwingend eine Grenze bei der Sentiment Analyse darstellen. Die Verfahren beweisen, dass sich Ironie und Sarkasmus erkennen und entsprechend analysieren lassen. In vielen Ansätzen zur Sentiment Analyse von Texten wird die Erkennung von Sarkasmus jedoch nicht berücksichtigt.[153]

Um jedoch ein umfassendes Verständnis der Sentiment Analyse, beziehungsweise des Opinion Minings, darzustellen, sollte jedoch auch der Begriff Opinion, also die Meinung, definiert werden. *Aggarwal und Zhai* definiert Opinion im Buch *Mining Text Data* wie folgt:

> „ An opinion (or regular opinion) is a quintuple, (e_i, a_{ij}, oo_{ijkl}, h_k, t_l), where e_i is the name of an entity, a_{ij} is an aspect of e_i, oo_{ijkl} is the orientation of the opinion about aspect a_{ij} of entity e_i, h_k is

[152] Vgl. Tsur, Davidov und Rappoport, 2010.
[153] Popescu und Etzioni, 2007; Pang und Lee, 2008.

> the opinion holder, and t_l is the time when the opinion is expressed by h_k. The opinion orientation oo_{ijkl} can be positive, negative or neutral, or be expressed with different strength/intensity levels."[154]

Bei dieser Definition handelt es sich um die Definition für eine reguläre Meinung. *Aggarwal und Zhai* unterscheidet zwischen regulären Meinungen und vergleichenden Meinungen, bei denen zwei oder mehr verschiedene Entitäten miteinander verglichen werden. Er fokussiert sich auf die regulären Meinungen. Damit ist immer eine einfache Meinung gemeint, die als positive oder negative Einstellung, eine Haltung oder eine Emotion gegenüber einer Entität durch den Meinungsäußerer dargestellt wird.[155] Auch *Pozzi u. a.* bedienen sich im Buch *Sentiment Analysis in Social Networks* dieser Definition von Opinion, da dies eine genauere Definition ermöglicht. Unter Wissenschaftlern ist umstritten, ob es Sentiment Analyse oder Opinion Mining heißen soll, da die Begriffe Sentiment und Opinion unterschiedlich definiert werden. Es hängen jedoch beide unweigerlich miteinander zusammen, da beide Elemente, die des Anderen enthalten.[156] Die im Zitat genannte Opinion orientation oo_{ijkl} stellt die dabei die sogenannte Sentimentpolarität dar, welches positiv, negativ oder neutral ausfallen kann. *Richard Heimann* beschreibt das Sentiment entsprechend:

Sentiment = data source, source, target, sentiment, polarity[157]

Nach *Richard Heimann* besteht das Sentiment unter anderem aus der Datenquelle, also woher, oder von wem, der analysierte Text stammt und ob es sich um einen Satz oder ein ganzes Dokument handelt. Außerdem enthält es die Quelle, die das Sentiment ausgedrückt hat und das Ziel, an das es gerichtet ist. Daneben ebenfalls der Typ der Emotion, also beispielsweise Liebe oder Hass und die Polarität (positiv, negativ, neutral).[158] Im Grunde versucht die Sentiment Analyse Emotionen, und eine Haltung, gegenüber bestimmten Entitäten, meist Produkten, Marken oder Personen, zu analysieren. Der Psychologie-Professor Klaus Scherer definiert Emotion im Buch *The Neuropsychology of Emotion* wie folgt:

[154] Aggarwal und Zhai, 2012, S. 418; Vgl. H. Liu, M.-A. Abbasi und Zafarani, 2014, S. 463.
[155] Vgl. Aggarwal und Zhai, 2012, S. 418.
[156] Vgl. Pozzi u. a., 2016, S. 1.
[157] Richard Heimann, 2014, S. 58.
[158] Vgl. Richard Heimann, 2014, S. 58.

> *„Emotions (sentiments) are episodes of coordinated changes in several components in response to external and internal events of major significance to the organism.“*[159]

Scherer beschreibt Emotionen als Episoden von Veränderungen verschiedener Komponenten, als Antwort auf wichtige interne oder externe Ereignisse. *Richard Heimann* beschreibt die Sentiment Analyse auf Basis dieser Definition als Intention diese Veränderungen zu verstehen, sie zu messen und mit einem Sentiment in Verbindung zu setzen.[160] Dabei nennt er zum besseren Verständnis Scherers „Typology of emotions“.[161] Dabei unterscheidet Scherer zwischen folgenden Typen:[162]

- **Emotion** beschreibt Scherer als kurze Bewertung eines größeren Events wie beispielsweise glücklich oder verärgert sein.
- **Mood** ist eine diffuse, weniger intensive Veränderung des subjektiven Gefühls. Als Beispiele nennt *Richard Heimann* depressiv oder fröhlich sein.
- **Interpersonal stance** ist eine gefühlsbedingte Haltung gegenüber einer anderen Person während einer bestimmten Interaktion, wie unter anderem distanziert sein oder freundlich sein.
- **Attitude** ist eine langfristige und gefühlsbedingte Haltung, oder Überzeugung, zu Objekten oder Personen. Dazu zählen das Lieben, Hassen oder Werten von etwas.
- **Personality traits** sind dauerhafte und typische Verhaltensweisen, wie nervös oder rücksichtslos zu sein.

Natürlich ist es immer abhängig vom Anwendungsfall in dem Sentiment Analyse eingesetzt werden soll, jedoch kann man, wie *Richard Heimann* sagt, in der Regel davon ausgehen, dass man über Scherers *Emotion* spricht. In manchen Anwendungsfällen wo Langzeitanalysen durchgeführt werden, können auch Moods eine wichige Rolle spielen.
Schieber, Hilbert und Stillich erwähnen in ihrem wissenschaftlichen Artikel eine Einteilung des Opinion Minings in drei verschiedene Aufgabenstellungen die *B. Liu* in seinem Buch *Web Data Mining - Exploring Hyperlinks,*

[159] Borod, 2000, S. 138 f.; Richard Heimann, 2014, S. 56.
[160] Richard Heimann, 2014.
[161] Vgl. Borod, 2000, S. 139 ff.
[162] Vgl. Richard Heimann, 2014, S. 56.

Contents, and Usage Data definiert.[163] *B. Liu* unterscheidet dabei zwischen diesen drei Aufgabenbereichen:

- **Sentiment classification**
- **Featured-based opinion mining and summarization**
- **Comparative sentence and relation mining**

Als *Sentiment classification* beschreibt *B. Liu* die Klassifizierung von Dokumenten, ob diese positiv oder negativ verfasst worden sind. Klassische Beispiele für diesen Aufgabenbereich sind die Klassifizierung von Bewertungen nach positiven und negativen Texten. Diese Klassifizierung geschieht meist auf Dokumentenebene und betrachtet dabei nicht die Details im Text, wie oder was Nutzer mochten oder eben nicht.

Unter *Featured-based opinion mining and summarization* versteht *B. Liu* den Ansatz, bei dem explizit auf die, in der Sentiment classification ausgelassenen, Details eingegangen wird. Dazu wird der Text auf Satzebene analysiert, um zu analyiseren, welche Eigenschaften einer Entität den Leuten gefallen oder nicht gefallen haben. Das Beispiel anhand einer Produktbewertung zeigt dies auf einfache Weise. Nimmt man den Satz "Das Handy bietet eine schlechte Auflösung des Displays", wäre das Ergebnis, dass der Autor sich auf die Auflösung des Displays bezieht und die Meinung dazu negativ sei. Das Ergebnis der Analyse kann dann genutzt werden, um beispielsweise explizit, bestimmte Eigenschaften von Produkten zu verbessern. Da die Analyse auf Satzebene erfolgt, können die Texte detaillierter analysiert, sowie die Beziehungen zwischen Objekten und der Meinung des Autors ausgewertet werden.

Als dritten Aufgabenbereich sieht *B. Liu* das *Comparative sentence relation mining*. Im Grunde ist der Kern der Analyse dabei das Vergleichen von einem Objekt mit einem oder mehreren anderen Objekten. Bei dieser Art der Analyse werden explizit vergleichende Sätze identifiziert und analysiert. Ein Beispiel dafür wäre der Satz „Der Akku des Smartphones X ist besser als der Akku des Smartphones Y“. Hier wird nicht die Meinung für ein Objekt alleine analysiert, sondern immer im Vergleich zu mindestens einem weiteren Objekt. Dabei spricht man von einer implizierten Meinung.[164]

[163] Vgl. Schieber, Hilbert und Stillich, 2012, S. 4; Vgl. B. Liu, 2007, S. 411.
[164] Vgl. Pozzi u. a., 2016, S. 7.

Pozzi u. a. zeigen in Abbildung 4.1 eine noch viel detailliertere Aufteilung der Sentiment Analyse in verschiedene Aufgabenbereiche.

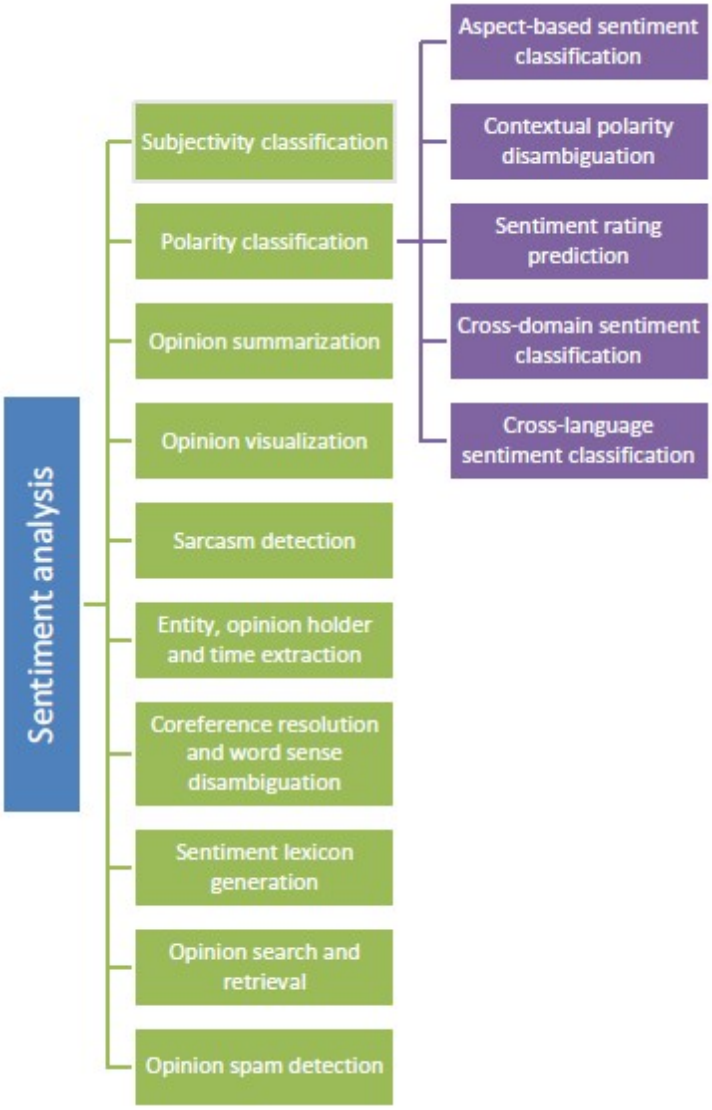

ABBILDUNG 4.1: Sentiment Analysis Aufgaben (Pozzi u. a., 2016, S. 4)

Die Abbildung stammt aus dem Buch *Sentiment Analysis in Social Networks* und deckt alle Bereiche der Sentiment Analyse ab und beinhaltet dabei beispielsweise auch die bereits angesprochene Identifizierung von Ironie und Sarkasmus. Die vorliegende Arbeit ist im Bereich der „Polarity classification" einzuordnen, da Kurztexte anhand ihrer Polarität (positiv, negativ, neutral) klassifiziert werden sollen. Im folgenden Abschnitt soll genauer auf die Ansätze zur Polaritätsklassifizierung eingegangen werden.

4.2 Methoden der Sentiment Analyse

Im Laufe der Zeit haben sich im Bereich der Sentiment Analyse verschiedene Ansätze entwickelt, die unter anderem dazu dienen, die Polarität zu klassifizieren.

4.2.1 Textverarbeitung und Natural Language Processing

Basis für die Durchführung einer Sentiment Analyse ist die Vorbereitung des Textes, bei der auch NLP-Techniken zur Anwendung kommen können. Unter Natural language processing versteht *Kao* folgendes: „Natural language processing, is the attempt to extract a fuller meaning representation from free text“[165]. Im Vergleich zum Beginn dieses Kapitels, wo der Text durch numerische Werte (bspw. Worthäufigkeit) repräsentiert wurde, versucht man durch NLP auch die Bedeutung des Textes zu berücksichtigen. Auch dabei gibt es verschiedene Formen, wie die natürliche Sprache, entsprechend beschrieben und / oder interpretiert werden kann. *Wilcock* stellt sieben Perspektiven dar um Sprache zu beschreiben(siehe Tabelle 4.2):

discourse	cohesion in a text ot dialogue
pragmatics	functions of utterances
semantics	meaning of words and sentences
syntax	word order and sentence structure
morphology	word formation and inflections
orthography	spelling (written language)
phonology	sounds (spoken language)

TABELLE 4.2: Levels of linguistic description (Wilcock, 2009, S. 12)

Hiermit können alle Aspekte der natürlichen Sprache untersucht und beschrieben werden. Syntax beispielsweise behandelt die Ordnung von Wörtern und den Aufbau von Sätzen, wohingegen Pragmatics die Funktionen der Aussprache behandelt. Dabei fällt auch auf, dass sich bestimmte Aspekte, auf die gesprochene und andere nur auf die geschriebene Sprache fokussieren. Für die Sentiment Analyse von Texten sind die Aspekte die sich mit der geschriebenen Sprache auseinandersetzen entscheidend, wie zum Beispiel Syntax, Morphology und Semantics.

[165] Vgl. Kao, 2006, S. 1.

Hofmann und Chisholm stellen in ihrem Buch *Text Mining and Visualization* einen fünfstufigen Prozess der Sentimentklassifizierung und Visualisierung dar.[166] Dieser beginnt mit der Datenbeschaffung und danach folgt die Datenverarbeitung, wobei in der Regel dann NLP-Techniken angewandt werden. Im Anschluß daran werden die Sentimente klassifiziert, die Klassifizierung evaluiert, um die Performance zu überprüfen und am Ende kann das Ergebnis visualisiert werden. Letzteres zählt jedoch dann schon zur Sentimentvisualisierung und nicht mehr direkt zur Sentimentklassifizierung. Im Folgenden sollen vor allem der Schritt der Verarbeitung von Texten und damit auch verschiedene NLP-Techniken näher betrachtet und erläutert werden.

Bei der Verarbeitung der Texte spielen unter anderem Smileys eine besondere Rolle. Diese drücken in der Regel besonders stark eine Emotion aus. Beispiele hierfür sind ":(" = traurig oder ":)" = glücklich. Damit diese später besser verarbeitet werden können, werden sie durch vorher definierte Strings ersetzt. Die nächste Aufgabe in Schritt 2 des Prozesses ist meist, einen Text in kleinere Bestandteile zu zerlegen. Das können Wörter oder Sätze sein. Dies nennt man *Tokenization*. Meist werden dazu Texte zuerst in Sätze heruntergebrochen und anschließend in einzelne Wörter. Die Tokens für die Sentiment Analyse basieren auf einem String aus einzelnen Buchstaben, ganzen Wörtern oder Wort-Kombinationen, den sogenannten *n-grams*.[167] Das Unterteilen des Textes in sogenannte Tokens kann am einfachsten erfolgen in dem nach Leerzeichen und / oder Punkten gesucht wird. Bei Tokens handelt es sich um sprachlich plausible Einheiten, in der Regel sind dies Wörter.[168] Jedoch gibt es auch hierbei Besonderheiten zu beachten, da der einfache Ansatz, lediglich nach Leerzeichen und Punkten zu trennen, schnell an seine Grenzen kommt. *Ananiadou und McNaught* beschreiben im Buch *Text mining for biology and biomedicine* verschiedene Punkte, die es zu beachten gilt. Ein wichtiger Punkt, der meist außer Acht gelassen wird, ist die Verwendung von Abkürzungen. Diese werden normalerweise mit einem Punkt abgekürzt. Hier gilt es, diese von einem Punkt an einem Satzende zu unterscheiden. Noch schwieriger wird dies, wenn die Abkürzung am Satzende steht. Da wird dann die sogenannte *Sentence boundary detection* wichtig. Dabei soll das Satzende identifiziert werden, häufig

[166] Vgl. Markus Hofmann, 2016, S. 135.
[167] Vgl. Markus Hofmann, 2016, S. 138.
[168] Vgl. Ananiadou und McNaught, 2006, S. 16.

anhand der gängigen Satzendungen Punkt, Ausrufezeichen oder Fragezeichen. Es kann jedoch auch vorkommen, dass Sätze mit anderen Zeichen beendet werden, wie Doppelpunkt oder Semikolon.[169] Dies muss dann entsprechend berücksichtigt werden. Auch Apostrophe und Bindestrich müssen richtig interpretiert werden. Dabei wird beispielsweise entschieden ob durch Bindestriche getrennte Wörter als ein oder mehrere Token interpretiert werden. Als Beispiel nennen *Weiss, Indurkhya und Zhang* eine Telefonnummer, wie 0800-319951. Bei dieser Nummer sollten die Zahlen vor und hinter dem Bindestrich nicht als unterschiedliche Token betrachtet werden, sondern die ganze Telefonnummer als ein Token.[170] Bei der Tokenization und der Sentence boundary detection handelt es sich um Aufgaben, die, der im Natural language processing, dem Aspekt Orthography zugeordnet werden können.[171] Im Anschluss an die Tokenization können auch sogenannte Stoppwörter und Satzzeichen entfernt werden. Unter Stoppwörtern versteht man Wörter, die häufig in Texten vorkommen und so gut wie keine Aussagekraft über das Dokument besitzen und auch für die Analyse meist kein Potenzial bieten.[172] In der deutschen Sprache sind das unter anderem Artikel wie „der", „die" oder „das".

Für die Sentiment Analyse kann es, gerade bei statistischen Ansätzen, vorteilhaft sein, wenn Wörter in einer generalisierten Form vorliegen. In der natürlichen Sprache werden Wörter häufig durch den grammatikalischen Kontext verändert. Zuerst werden alle Wörter kleingeschrieben, sprich aus „Haus wird „haus". Im Anschluss daran werden die Wörter in eine generische Form gebracht. Dies nennt man in der Literatur *morphologische Analyse*[173], *Stemming* oder auch *Lemmatization*[174]. Das Ziel dabei ist es, die Wörter in ihre Grundform zu bringen, das sogenannte Lemma. Der Begriff morphologische Analyse stammt aus der Linguistik und steht für die Transformation von Wörtern, bzw. deren grammatikalisch bedingten Varianten, zur Grundform. Dies wird im Rahmen der Text Analyse auch als *Inflectional Stemming* bezeichnet.[175] Das ist darauf zurückzuführen, dass man die Veränderung eines Wortes um grammatikalische Bedingungen wiederzugeben, auch Flexion nennt. Die flektierten Worte sind genau die, die in ihre Grundform gebracht werden müssen. Ob Stemming durchgeführt wird, ist

[169] Vgl. Ananiadou und McNaught, 2006, S. 16 f.
[170] Vgl. Weiss, Indurkhya und Zhang, 2015, S. 17.
[171] Vgl. Wilcock, 2009, S. 19.
[172] Vgl. Richard Heimann, 2014, S. 38; Vgl. Weiss, 2004, S. 26 f.
[173] Vgl. Ananiadou und McNaught, 2006, S. 18.
[174] Vgl. Ananiadou und McNaught, 2006, S. 19; Vgl. Markus Hofmann, 2016, S. 140 f.
[175] Vgl. Weiss, Indurkhya und Zhang, 2015.

immer abhängig von den geplanten Anwendungen, jedoch kann es gerade bei Klassifizierungen einen Vorteil bringen. Es hilft die Analyse zu vereinfachen, da es weniger verschiedene einzigartige Typen im Text gibt. In einem Beispiel könnten die Wörter „Wortes“, „Wörter“ und „Wort“ in einem Text auftreten. Ohne Stemming wären die drei Wörter drei verschiedene einzigartige Typen, obwohl der Wortstamm immer der gleiche ist. Mit Stemming würden diese Wörter auf den Wortstamm „Wort“ zusammengefasst werden und nur ein einzigartiger Typ aber mit einer Frequenz von drei. Die Komplexität des Stemmings hängt auch immer mit der Sprache des Textes zusammen, da Wörter in verschiedenen Sprachen unterschiedlich flektiert werden. *Weiss, Indurkhya und Zhang* nennen Spanisch als vergleichsweise einfache Sprache, wohingegen Englisch und Deutsch zu den komplexeren gehören.[176] Dies ist vor allem auf häufige irreguläre Wortformen zurückzuführen. In der deutschen Sprache ist es auch das Einfügen von Buchstaben innerhalb eines Wortes, beispielsweise das Wort „angeben“ und die Vergangenheitsform „angegeben“.

In der Literatur und Wissenschaft finden sich jedoch auch unterschiedliche Ansätze um Stemming durchzuführen. Dabei gibt es einmal lexikonbasierte und nicht-lexikon-basierte Ansätze. Liest man über nicht-lexikonbasierte Ansätze, wird dabei meist der Porter Algorithmus erwähnt. Dieser ist nach seinem Erfinder Martin Porter 1980 entwickelt und im Artikel „An Algorithm for Suffix Stripping“ vorgestellt worden. Das Ziel des Algorithmus ist, automatisch Suffixe in Wörtern zu entfernen. Der Algorithmus arbeitet dabei mit vorher definierten Regeln. Er wurde von Porter für die englische Sprache entwickelt, kann jedoch, nach Anpassung, auch für andere Sprachen eingesetzt werden. Die definierten Regeln dienen dazu, das Wort zu verkürzen. Dabei werden die Regeln solange auf ein Wort angewendet, bis eine Minimalanzahl an Vokal-Konsonant-Sequenzen erreicht ist. Dabei kann jedes Wort oder jeder Teil eines Wortes, als Kette von Zeichen in der Form $[C](VC)^{\mathrm{m}}[V]$ dargestellt werden.[177] Dabei dient *m* dazu, die Anzahl an Vokal-Konsonant-Sequenzen zu messen. Der Konsonant *C*, zu Beginn der Kette, ist optional, genau wie eine Folge von Vokalen *V*, am Ende des Wortes. Ein Wort wird immer an eine Gruppe von Regeln gegeben, von welcher immer nur eine Regel angwandt wird und zwar die mit der größten Übereinstimmung. Die Regeln sind immer nach dem Muster $(Kondition)S1 \rightarrow S2$ (S = Suffix) aufgebaut. Um zu demonstrieren, wie dies

[176] Vgl. Weiss, Indurkhya und Zhang, 2015, S. 19.
[177] Porter, 1997, S. 214.

funktioniert, nutzt *Porter* folgende Gruppe mit Regeln:[178]

SSES $\rightarrow$ SS

IES $\rightarrow$ I

SS $\rightarrow$ SS

S $\rightarrow$

Bei diesen Regeln sind der Einfachheit halber keine Konditionen vorhanden. Als Beispiel nennt *Porter* unter anderem das Wort CARESSES. Die größte Übereinstimmung hat das Wort mit der Regel für den Suffix S1 = SSES. Entsprechend wird aus dem Wort CARESSES das Wort CARESS. Das Wort CARES stimmt mit der letzten Regel der Gruppe überein, da S1 = S und wird entsprechend zu CARE umgewandelt. Ein Beispiel für eine Regel mit einer Kondition ist $(m > 0)EED \rightarrow EE$. Wendet man diese Regel auf das Wort FEED an, so ist das Ergebnis FEED, da m = 0. Der einfache Aufbau, seine Performance und seine Anpassungsfähigkeit machen den Porter Algorithmus populär, wenn es um Stemming geht, das nicht lexikonbasiert ist. Natürlich ist der Algorithmus nicht zu hundert Prozent genau, da es immer wieder Fälle gibt, die nicht korrekt verarbeitet werden. Außerdem kann nicht immer davon ausgegangen werden, dass die Wortstämme, die durch den Algorithmus abgeleitet werden linguistisch korrekt sind. Dies ist darauf zurückzuführen, dass verwandte Worte nur auf eine gemeinsame Zeichenkette zusammengeführt werden. Trotzdem sind seine Geschwindigkeit, die durchaus guten Ergebnisse und die vergleichsweise einfache Portierbarkeit des Algorithmus in verschiedene Sprachen, große Vorteile.

Neben dem lexikonfreien Ansatz wird meist der lexikonbasierte Ansatz eingesetzt. Dabei existieren zwei verschiedene Varianten des lexikonbasierten Ansatzes. Bei einer der Varianten kommen sogenannte „full-form lexicons“[179] zum Einsatz. Bei dieser Variante handelt es sich um die vergleichsweise simple Form der lexikonbasierten Ansätze. Unter „full-form lexicon“ versteht man ein Wörterbuch, dass alle morphologischen Varianten eines Wortes inklusive seiner grammatikalischen Merkmale, enthält. Dadurch kann die Grundform eines jeden Wortes egal in welcher grammatikalischen Form es vorliegt, einfach nachgeschlagen werden.[180] Der Aufwand, die Grundform zu bestimmen, ist bei diesem Ansatz vergleichsweise simpel, da die Form

[178] Vgl. Porter, 1997, S. 215.
[179] Vgl. Ananiadou und McNaught, 2006, S. 18.
[180] Vgl. Ananiadou und McNaught, 2006, S. 18.

nur nachgeschlagen und nicht abgeleitet werden muss.

Die andere Variante des lexikonbasierten Ansatzes baut auf ein Lexikon mit Grundformen für Wörter auf. Außerdem werden dabei Listen, beispielsweise für die Flexion oder Ableitung von Wörtern, relevanten Suffixe genutzt. Die Analyse läuft dann so ab, dass beide Quellen kombiniert eingesetzt werden. Zuerst wird geschaut, ob eine bestimmte Zeichenkette (z.B. ein Suffix) im Wort enthalten ist. Dann wird diese aus dem Token entfernt. Der daraus resultierende Token wird dann mit den Grundformen im Lexikon abgeglichen. Wird kein passender Eintrag gefunden, wird Schritt eins erneut auf den Token angewandt, bis eine passende Grundform gefunden wird oder der Prozess anderweitig unterbrochen wird, weil gegebenenfalls kein passender Eintrag im Lexikon existiert.[181]. *Ananiadou und McNaught* nennen, als methodisch fortgeschrittensten Ansatz den, dass man verschiedene Wandler parallel betreibt, die jeder ein anderes morphologisches Phänomen berücksichtigen, im Englischen beispielsweise, das Ändern von *y* zu *ies*, wie bei *fl-y* und *fl-ies*.[182]

Eine weitere Anwendung aus dem Bereich des Natural language processing stellt das sogenannte *Part-of-Speech-Tagging* dar, welches ebenfalls in den Bereich der Morphologie einzuordnen ist. Dies kann beispielsweise nach der Tokenization angewendet werden, um die einzelnen Teile des Textes bestimmten Satzteilen zuzuordnen. Dabei werden die einzelnen Token grammatikalischen Klassen zugeordnet, beispielsweise ob es sich um ein Nomen oder ein Verb handelt. Die Anzahl der grammatikalischen Klassen unterscheidet sich zwischen verschiedenen Sprachen stark. In manchen Sprachen gibt es bis zu 100 verschiedene Klassen.[183] Natürlich ist es auch immer abhängig vom verwendeten Tagger, da dieser natürlich auch granularer klassifizieren kann. Dies kann beispielsweise sein, dass ein Verb nicht nur als solches deklariert wird, sondern zum Beispiel auch, in welcher Zeitform sich das Verb befindet. Das Problem beim Tagging ist meist auch, dass Wörter verschiedene Bedeutungen haben können und sowohl als Nomen, als auch als Verb auftreten können. Trotzdem erreichen Part-of-Speech Tagger häufig eine sehr gute Performance, wie beispielsweise *Cutting u. a.* in ihrem wissenschaftlichen Artikel „A practical part-of-speech tagger“ mit circa 96% Genauigkeit, was ein beachtlicher Wert ist, wenn man bedenkt, dass viele

[181] Vgl. Ananiadou und McNaught, 2006, S. 18 f.
[182] Ananiadou und McNaught, 2006, S. 19.
[183] Vgl. Weiss, Indurkhya und Zhang, 2015, S. 31.

Wörter in mehrere Klassen eingeordnet werden können.[184] *Manning* zeigt im Buch *Foundations of Statistical Natural Language Processing* wie solche Probleme behandelt werden können. Dazu zieht man beim Taggen, unter anderem Informationen über gängige syntaktische Satzkonstellationen hinzu. In der deutschen Sprache wäre ein gängiger Fall Pronomen - Verb - Artikel - Nomen, wie zum Beispiel „Ich esse ein Eis." oder „Wir schreiben einen Text.". Anhand der Sequenz kann dann eine Entscheidung getroffen werden, ob es sich bei dem Wort um beispielsweise ein Verb oder Nomen handelt.[185] Dies wird auch syntagmatischer Ansatz genannt.[186] Anhand dieser Regel haben *Greene und G. Rubin* im Jahr 1971 eine Genauigkeit von 76% erreicht.[187] Einen lexikalischen Ansatz haben *Charniak u. a.* verfolgt, da sie einfach für ein Wort den wahrscheinlichsten Tag vergeben haben, wodurch sie eine Genauigkeit von ca. 90% erreicht haben.[188] Um die Genauigkeit noch weiter zu steigern, wird bei sehr erfolgreichen Ansätzen meist eine Kombination aus syntagmatischem und lexikalischem Vorgehen verwendet. Alternativ können auch, wie bei *Cutting u. a.*, lernende Algorithmen mit einem lexikalischen Ansatz verbunden werden. Damit erreichten sie eine Genauigkeit von 96%.[189]

Bei den hier vorgestellten NLP-Techniken handelt es sich nur um eine Auswahl aus dem Bereich des Natural Language Processing. Dabei sind jedoch auch Anwendungen, die nicht in den Rahmen dieser Ausarbeitung passen, wie beispielsweise Machine Translation, also die automatisierte Übersetzung von einer Sprache in eine andere, ebenfalls unter NLP einzuordnen ist.

4.2.2 Lexikonbasierter Ansatz

Den vergleichsweise simpelsten Ansatz zur Sentimentklassifizierung stellt der lexikonbasierte Ansatz dar. Dabei werden, ähnlich wie bei dem Stemming-Prozess, Lexika verwendet. Der Unterschied zu den Stemming-Lexika ist, dass nicht Grundformen nachgeschlagen werden, sondern Sentimente. Dies erfolgt indem beispielsweise das Wort „gut" nachgeschlagen wird und dem Wort *gut* im Lexikon das Sentiment *positiv* zugeordnet ist, welches dann zurückgegeben wird. Das gleiche wäre im umgekehrten Sinne bei dem Wort

[184] Vgl. Cutting u. a., 1992, S. 1.
[185] Vgl. Manning, 1999, S. 343.
[186] Vgl. Manning, 1999, S. 344 f.
[187] siehe Greene und G. Rubin, 1971; Vgl. Manning, 1999, S. 343.
[188] Vgl. Charniak u. a., 1993, S. 1 ff.
[189] Vgl. Cutting u. a., 1992, S. 1.

„schlecht“ der Fall. Dabei wird im Lexikon für das Wort *schlecht* das Sentiment *negativ* ausgegeben. In *Sentiment Analysis in Social Networks* werden als Beispiele das *Subjectivity Lexicon* oder das *Opinion Lexicon* genannt.[190] Letzteres beinhaltet rund 6800 englische Wörter mit dem jeweils dazugehörigen Sentiment (positiv oder negativ). Das Subjectivity Lexicon wurde im wissenschaftlichen Artikel „Recognizing Contextual Polarity in Phrase-level Sentiment Analysis“ von *Wilson, Wiebe und Hoffmann* genutzt[191] und enthält rund 8200 englische Wörter, inklusive einem Kennzeichen, ob ein Wort in den meisten Kontexten subjektiv ist oder meistens nicht oder wenig subjektiv ist. Daneben enthält es auch ein sogenanntes *Part-of-Speech*-Kennzeichen, wodurch ersichtlich wird, ob es sich zum Beispiel um ein Verb oder Nomen handelt. Diese beiden Lexika zeigen jedoch auch bestimmte Nachteile auf, die mit der Benutzung von Lexika bei der Sentimentklassifizierung auftreten können. Meist werden in einem Lexikon die Sentimente klar bestimmt. Das bedeutet, dass es nur positiv, negativ oder neutral gibt und keine Skala für Positivität oder Negativität. *Pozzi u. a.* sehen außerdem zusätzlich, dass ein Sentiment mit einem Wort, anstatt mit dem Sinn des Wortes, assoziiert wird.[192] Als Beispiel wird das englische Wort „crazy“ aufgezeigt. Das Wort kann sowohl positiv, als auch negativ gemeint sein, da es immer abhängig vom Kontext ist, in dem es genutzt wird.[193] Dadurch ist es in vielen Anwendungsfällen schwer, eine einfache Liste von Wort-Sentiment-Paaren zu benutzen, um eindeutig und korrekt das Sentiment eines Textes zu bestimmen.

Natürlich haben sich die Ansätze und Lexika mit der Zeit verbessert, so gibt es auch Lexika, die nicht hart ein Sentiment bestimmen, sondern einen sogenannten *Sentiment* oder *Polarity Score* zu jedem Wort ausweisen. Würde man hier wieder die Wörter „gut“ und „schlecht“ als Beispiele heranziehen, könnte *gut* einen Score von positiv vier haben und *schlecht* einen Score von negativ vier. Für jedes Wort im Text wird dann ein Score nachgeschlagen und die Werte in der Regel aufsummiert, um einen Gesamtscore zu bilden. Der Satz „Der Film war gut!“ hätte demnach einen Gesamtscore von positiv vier, da die ersten drei Wörter neutral einzustufen sind und *gut* einen positiven Score von vier hat. Ebenfalls gibt es Ansätze, die sich mit

[190] Vgl. Pozzi u. a., 2016, S. 32.
[191] siehe Wilson, Wiebe und Hoffmann, 2005.
[192] Vgl. Pozzi u. a., 2016, S. 32 f.
[193] Vgl. Pozzi u. a., 2016, S. 33.

dem Sinn der Wörter beschäftigen, wie beispielsweise SentiWordNet. SentiWordNet basiert dabei auf WordNet[194], einer Datenbank, die dafür entwickelt wurde Texte für Computer verständlich zu machen und enthält entsprechend, die semantischen Beziehungen zwischen Wörtern. SentiWordNet erweitert WordNet um einen Sentimentwert, der für unterschiedliche semantische Bedeutungen des Wortes auch verschieden ausfallen kann.[195] Dabei setzt sich der Sentimentwert aus drei verschiedenen Werten zusammen, der jeweils aussagt, wie positiv, negativ oder neutral ein Wort ist. Die Werte bewegen sich in dem Intervall *[0.0, 1.0]* und ergeben in der Summe immer *1.0*. Dies kann auch bedeuten, dass ein Wort auch Werte größer als null, in jedem der drei Kategorien, aufweist. *Baccianella, Esuli und Sebastiani* nehmen in ihrem wissenschaftlichen Artikel „SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining." zur Erklärung das Wort „estimable". Das Wort wird im WordNet mit verschiedenen semantischen Bedeutungen aufgelistet. Um zu demonstrieren wie unterschiedlich die Werte für verschiedene Bedeutungen eines Wortes ausfallen können, zeigen sie dies anhand zwei verschiedener Bedeutungen des Wortes. Für die Bedeutung „may be computed or estimated" hat das Wort einen neutralen Wert von *1.0* und die Werte für positiv und negativ sind jeweils *0.0*. Wohingegen die Bedeutung „deserving of respect or high regard" einen positiven Wert von *0.75*, einen negativen von *0.0* und einen neutralen Wert von *0.25* aufweist.[196] Hier wird schon deutlich, dass die Bedeutung des Wortes entscheidend für die Bewertung eines Sentiments sein kann. Estimable, mit der ersten Bedeutung, ist gänzlich neutral anzusehen, wohingegen die andere Bedeutung deutlich überwiegend positiv zu bewerten ist. Diese Art von Lexikon wird auch „sense-aware lexicon" genannt.[197]

Unterschiedliche Ansätze bei Lexika, zum Beispiel mit einem Sentimentwert oder nur mit einer Polarität, können natürlich auch immer zu unterschiedlichen Ergebnissen führen. Beispielsweise bei Lexika mit einem Sentimentwert, wo Wörter in dem einen Lexikon anders eingestuft sind als in einem anderen. Wörter könnten eher als neutral gewertet werden oder bei dem einen sind Wörter neutraler eingestuft, wohingegen sie jedoch mehr polar sind.[198] Das ist darauf zurückzuführen, dass viele Lexika manuell und

[194] siehe Fellbaum, 1998.
[195] Vgl. Baccianella, Esuli und Sebastiani, 2010, S. 1.
[196] Vgl. Baccianella, Esuli und Sebastiani, 2010, S. 2200.
[197] Vgl. Pozzi u. a., 2016, S. 33.
[198] Vgl. Richard Heimann, 2014, S. 60.

häufig zur Lösung eines bestimmten Problems, beziehungsweise Anwendungsfalls, erstellt wurden. *Richard Heimann* beschreibt vorgefertigte Lexika als guten Startpunkt, jedoch nicht als umfassende Lösung. Wie bereits erwähnt, werden die Lexika meist für einen bestimmten Anwendungsfall erstellt und sind damit häufig spezifisch für ein bestimmtes Problem oder eine Domäne. Aufgrund dessen sieht *Richard Heimann* vorgefertigte Lexika nur als Startpunkt, da diese meist schon vergleichsweise gut funktionieren, doch in der Regel nicht auf das eigene Problem zugeschnitten sind. Entsprechend sollte man immer Lexika entwickeln, die anwendungsdomänenspezifisch sind, um bestmögliche Ergebnisse zu erhalten. Die bereits bestehenden Lexika können dann der Ausgangspunkt sein und dem Problem entsprechend erweitert oder angepasst werden. Andernfalls kann das Lexikon auch komplett selbst angefertigt werden.

Ebenfalls eine Fehlerquelle bei der Analyse von Texten kann die Missachtung des Kontextes sein. Wird beispielsweise eine Filmkritik analysiert und der Autor beschreibt darin eine düstere Filmszene, kann diese als negativ eingestuft werden, obwohl die Filmkritik an sich positiv über den Film geschrieben ist. Dies kann auftreten, wenn der Kontext des Textes nicht beachtet wird und ist meist anwendungsspezifisch. Werden Filmkritiken analysiert, muss davon ausgegangen werden, dass auch Filmszenen beschrieben werden. In anderen Anwendungsfällen kommt so etwas eventuell gar nicht vor und kann dann auch vernachlässigt werden. Den lokalen Kontext mit einzubeziehen, ist mit lexikon-basierten Ansätzen einfacher umzusetzen, im Vergleich zu den Klassifizierungsmodellen, die im nächsten Abschnitt dieses Kapitels näher behandelt werden. So können beispielsweise Verneinung, wie „nicht gut“ oder Verstärkungen von Wörtern wie „sehr gut“, mit in die Betrachtung einfließen. *Polanyi und Zaenen* nennen dies *contextual valence shifters*.[199] *Kennedy und Inkpen* zeigen in ihrem Artikel „Sentiment classification of movie reviews using contextual valence shifters“, dass sich durch den Einbezug von *valence shifters* eine höhere Genauigkeit bei der Bestimmung der Sentimente in Texten erzielen lässt.

4.2.3 Statistische Verfahren / Machine Learning

Neben dem lexikonbasierten Ansatz existieren auch Machine Learning Ansätze, die in diesem Abschnitt näher betrachtet werden. Dabei unterscheidet

[199] Vgl. Polanyi und Zaenen, 2006.

man grundsätzlich zwischen zwei Arten, dem sogenannten *supervised learning* und dem *unsupervised learning*. Bei Ersterem handelt es sich um eine Art überwachtes Lernen, wobei ein Algorithmus angelernt wird. Dies geschieht normalerweise mit Hilfe eines Trainingssets an Daten, welches Eingabedaten und die dazugehörigen gewünschten Ausgabedaten enthält. Der Algorithmus versucht dann, auf Basis der Trainingsdaten, Muster zu definieren, die dann auf Nicht-Trainingsdaten angewendet werden können und den gewünschten Ausgabewert ausgeben.[200] Als Beispiel dafür nennt *Alpaydin* in seinem Buch *Introduction to Machine Learning* das Anlernen einer Klasse „Familienauto".[201] Als Trainingsset werden Autos danach kategorisiert, ob sie ein Familienauto sind oder nicht. Die Autos, die als Familienauto bestimmt werden, nennt man auch positive Beispiele, Autos, die nicht in die Kategorie fallen, negative Beispiele beinhalten. Das Anlernen funktioniert dann so, dass eine Beschreibung gesucht wird, die alle positiven Beispiele gemeinsam haben und keines der negativen Beispiele. Anhand dieser Beschreibung können dann, bisher unbekannte Autos entsprechend klassifiziert werden.

Beim nicht überwachten Training hingegen wird der Algorithmus nicht mit Hilfe von Trainingsdaten angelernt, sondern es wird versucht ein Muster aus den Daten abzuleiten. Der Algorithmus versucht dabei, nur anhand der vorhandenen Daten, Muster zu erkennen, ohne dass es eine bereits festgelegte Ausgabe gibt.[202] Dabei betrachtet der Algorithmus beispielsweise die Ähnlichkeit zwischen Objekten, um Rückschlüsse auf eine Klassifizierung machen zu können.[203]

Machine-Learning-Ansätze werden auch häufig in der Sentiment Analyse angewendet und wurden in der Wissenschaft kontinuierlich weiterentwickelt. Einer der ersten Wissenschaftler, der Machine Learning zur Sentiment Analyse eingesetzt hat, war *Turney* in seinem Artikel „Thumbs Up or Thumbs Down?: Semantic Orientation Applied to Unsupervised Classification of Reviews".[204] Dabei setzt er auf einen nicht überwachten Ansatz, um Bewertungen im Internet als *empfehlenswert* oder *nicht empfehlenswert* zu klassifizieren. Ob eine Bewertung gut oder schlecht ist, bewertet er, indem er den sogenannten *Pointwise Mutual Information* berechnet. Dabei

[200] Vgl. Sammut und Webb, 2011, S. 941.
[201] Vgl. Alpaydin, 2014, S. 21 f.
[202] Vgl. Sammut und Webb, 2011, S. 1009.
[203] Vgl. Mueller und Massaron, 2016, S. 169.
[204] siehe Turney, 2002.

handelt es sich um einen statistischen Koeffizienten, mit dem er versucht, die textuelle Nähe des einzuordnenden Wortes, zu den Worten „poor“ und „excellent“, gegeneinander aufzuwiegen.[205] Die textuelle Nähe versuchte er damals über die Suchmaschine AltaVista zu bestimmen, indem er dort versuchte die Anzahl an Dokumenten zu ermitteln, wo der einzuordnende Begriff zusammen mit dem Wort poor beziehungsweise excellent auftrat.[206]

Pang, Lee und Vaithyanathan nutzen in „Thumbs Up?: Sentiment Classification Using Machine Learning Techniques“ ebenfalls Machine Learning, um Texte in die beiden Sentimente positiv oder negativ zu klassifizieren. Dabei setzen sie jedoch auf die überwachte Lernmethode. *Pang, Lee und Vaithyanathan* weisen daraufhin, dass zu diesem Zeitpunkt (Jahr 2002) Machine Learning Experten der Sentiment Analyse durch automatisierte Verfahren keine großen Erfolgsaussichten prophezeiten, weil sie denken, dass die Intuition des Menschen treffsicherer sei als Algorithmen.[207] Um diese Aussage zu überprüfen, haben sie zwei Stundenten, unabhängig voneinander, beauftragt, jeweils eine Liste von positiven und negativen Wörtern zu definieren, die ihrer Meinung nach in entsprechenden Filmbewertungen vorkommen. Anhand dieser Liste wurden weitere Filmbewertungen klassifiziert. Dabei lag die Genauigkeit bei nur 58% und 64%. Daraufhin erstellten *Pang, Lee und Vaithyanathan* eine Liste mit jeweils positiven und negativen Wörtern, anhand ihrer Häufigkeit in den Testdaten und einer eigenen Überprüfung der daraus resultierenden Wörter. Die Liste aus dem automatisierten Verfahren lieferte bei der Klassifizierung bereits 69% und war damit bereits besser als die von Menschen erstellte Liste.[208] Um zu überprüfen, wie gut Machine Learning Algorithmen funktionieren, testeten sie drei verschiedene Algorithmen. Dabei handelt es sich um den Naive-Bayes-Klassifikator, den Maximum-Entropy und den Support-Vector-Machines Algorithmus. Mit diesen Algorithmen erreichten sie eine Genauigkeit bei der Klassifizierung zwischen knapp 79% und 83%.[209] Im Folgenden soll der Naive-Bayes-Klassifikator kurz vorgestellt werden.

Der Naive-Bayes-Klassifikator basiert auf dem Bayes Theorem.[210] Dabei wird im Bezug auf die Sentimentklassifizierung versucht, einem Dokument oder

[205] Vgl. Ziegler, 2006a, S. 108.
[206] Vgl. Ziegler, 2006a, S. 108; Vgl. Turney, 2002, S. 417 f.
[207] Vgl. Pang, Lee und Vaithyanathan, 2002, S. 2.
[208] Vgl. Pang und Lee, 2008, S. 3.
[209] Vgl. Pang, Lee und Vaithyanathan, 2002.
[210] Vgl. Bättig, 2014, S. 116.

Text *d* eine Klasse *c* (positiv, negativ), anhand der Wahrscheinlichkeit, zuzuweisen. Dies kann dann wie folgt ausgedrückt werden:

$$P(c|d) = \frac{P(d|c) * P(c)}{P(d)} \tag{4.1}$$

Daraus folgt, dass für ein gegebenes Dokument *d* die Klasse *c* durch $c = \arg\max_c P(c|d)$ zugewiesen wird. Diese Formel sagt vereinfacht aus, dass dem Dokument die Klasse mit der höchsten Wahrscheinlichkeit zugeordnet wird, da das Maximum für $P(c|d)$ gesucht wird. Beim Naive-Bayes-Klassifikator wird *d* durch seine Attribute $a_1, \ldots, a_n$ definiert oder dargestellt. Der Naive-Bayes-Algrithmus verfolgt die vereinfachte Annahme, dass alle Attribute unabhängig voneinander sind, daher auch das Wort „Naive".[211] Dies bedeutet, dass die Wahrscheinlichkeit eines Attributs unabhängig von den anderen Attributen zu betrachten ist. Natürlich kann diese Annahme nicht immer zutreffend sein und trotzdem erweist sich der Naive-Bayes-Klassifikator als häufig gute Lösung, da meist trotzdem gute Ergebnisse erzielt werden, solange die Attribute wenig voneinander abhängen. Daraus ergibt sich dann:

$$c = \arg\max P(c_j) * \prod P(a_i|c_j) \tag{4.2}$$

Da die Attribute unabhängig sind, ergibt sich die Wahrscheinlichkeit für die Gesamtheit der auftretenden Attribute aus dem Produkt der Wahrscheinlichkeiten der einzelnen Attribute. Im Lernschritt für das überwachte Anlernen des Naive-Bayes-Klassifikators werden $P(c_j)$ und $P(a_i|c_j)$ anhand der Häufigkeit im Trainingsdatenset geschätzt. Sobald der Klassifikator angelernt ist, kann, mit Hilfe der in der Lernphase ermittelten Wahrscheinlichkeiten und der Formel 4.2, ein unbekanntes Dokument *d* klassifiziert werden. Aufgrund der Tatsache, dass der Naive-Bayes-Klassifikator auf einer vergleichsweise einfachen Formel basiert, erreicht er im Machine Learning-Umfeld immer eine sehr gute Performance bei relativ guten Ergebnissen, solange die Attribute nicht zu stark korrellieren.

4.3 Evaluierung der Zuverlässigkeit

Ist beispielsweise das Training des Klassifikators abgeschlossen, dann ist es wichtig, die Performance zu evaluieren, um bestimmen zu können, wie genau und zuverlässig die Analyse funktioniert. Darauf werden dann auch Rückschlüsse auf die zukünftige Performance der Analyse geschlossen. Vor

[211] Vgl. Garreta und Moncecchi, 2013, S. 33.

allem bei Machine-Learning-Ansätzen versucht man, mit Hilfe eines Trainings- und eines Testdatensets, die Performance zu evaluieren. Dazu werden aus der Menge der Daten ein zufälliges Sample, also eine Stichprobe, für das Trainingsset und ein Weiteres für das Testset genommen.[212] Vor allem im Big Data-Umfeld ist die Evaluierung der Zuverlässigkeit unerlässlich, da man hier meist mit sehr großen Datenmengen arbeitet und noch präzisere Vorhersagen erwartet. Dabei ist die Annahme, dass mit mehr verfügbaren Daten, bei angelernten Modellen, weniger Probleme beim Test mit neuen Daten auftreten, nicht grundsätzlich richtig. Neben steigender Komplexität können auch sehr selten auftretende Probleme zu Problemen führen, da diese nur in sehr großen Stichproben entdeckt werden können.[213] Eine gute Methode um aus einem begrenzten Datenset verschiedene Stichproben zu testen, ist die sogenannte *k-fold Cross validation*. Bei der 4-fold Cross validation beispielsweise wird das Datenset zufällig in vier, möglichst gleich große Teile, eingeteilt.

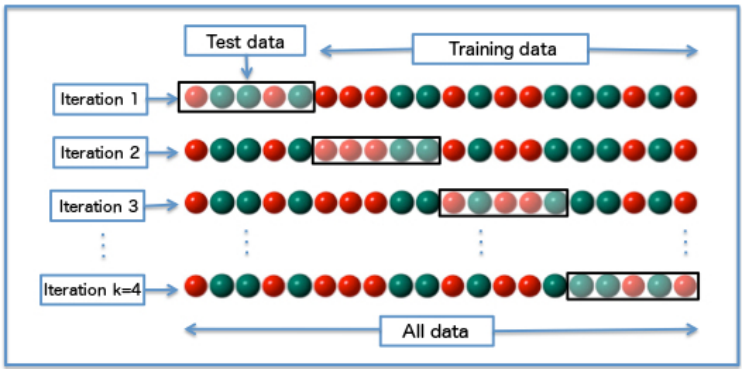

ABBILDUNG 4.2: 4-fold Cross validation (Quelle: http://bit.ly/2nB7IDW)

In Abbildung 4.2 ist der Vorgang beispielhaft dargestellt. Die Zahl Vier steht dabei für vier Iterationen der Validierung. Dabei rotiert das Testdatenset bei jeder Iteration durch das Gesamtdatenset. So können bei einem begrenzten Datenset mehr Daten für das Testen des Algorithmus verwendet werden, da das Trainings- und Testset, zufällig und k entsprechend, häufig wechselt.[214]

Neben der Kreuzvalidierung existieren auch bestimmte Metriken, um die Zuverlässigkeit und Genauigkeit eines lernenden Algorithmus zu bestimmen. Häufig trifft man im Bereich der Sentiment Analyse auf die sogenannte *Confusion Matrix* wie in Abbildung 4.3 zusehen.

[212] Vgl. Weiss, Indurkhya und Zhang, 2015, S. 69.
[213] Vgl. Weiss, Indurkhya und Zhang, 2015, S. 72.
[214] Vgl. Weiss, Indurkhya und Zhang, 2015, S. 72.

		Actual Sentiment	
		Positive	Negative
Predicted Sentiment	Positive	$TruePositive$	$FalsePositive$
	Negative	$FalseNegative$	$TrueNegative$

ABBILDUNG 4.3: Confusion Matrix (Markus Hofmann, 2016, S. 144)

Dabei stellen *True Positive* (TP) und *True Negative* (TN) die Anzahl dar, die korrekt klassifiziert wurde, wohingegen *False Positive* (FP) und *False Negative* (FN) die Anzahl angeben, die jeweils falsch zugeordnet wurden. Aus diesen Werten lässt sich dann die Genauigkeit für positive und negative Klassifizierungen errechnen.[215] Dabei werden gängige Metriken, wie beispielsweise *Recall*(Formel 4.4), *Precision* (Formel 4.3) und die sogenannte *F-Measure* (Formel 4.5), genutzt, um die Genauiget und Zuverlässigkeit zu bestimmen. In den aufgeführten Formeln ist dies anhand der Positivklasse veranschaulicht.

$$Precision = \frac{TruePositive}{TruePositive + FalesNegative} \tag{4.3}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{4.4}$$

$$F - Measure = \frac{2}{1/Recall + 1/Precision} \tag{4.5}$$

Bei der Precision, also der Genauigkeit, betrachtet man das Verhältnis zwischen den richtig Vorhergesagten und der Gesamtheit der Vorhergesagten für diese Klasse. Recall hingegen beschreibt das Verhältnis zwischen den richtig Vorhergesagten und der tatsächlich Richtigen, also den True Positive und False Negative.[216] Bei F-Measure handelt es sich um das gewichtete harmonische Mittel von Precision und Recall. Je nachdem welchem der beiden Kennzahlen man mehr Wichtigkeit zuschreibt, kann das harmonische Mittel entsprechend gewichtet werden. Abhängig von der Anwendung muss entschieden werden, ob Precision oder Recall die wichtigere Kennzahl ist. Bei sehr großen Datenmengen wird meist Wert auf eine hohe Genauigkeit gelegt. Dies bedeutet im Umkehrschluss, dass die Klassifizierung, falls etwas zu dieser Klasse zugeordnet wird, mit hoher Wahrscheinlichkeit korrekt ist. Jedoch können auf der anderen Seite nicht alle Korrekten erfasst werden,

[215] Vgl. Markus Hofmann, 2016, S. 144 f.
[216] Vgl. Sammut und Webb, 2011, S. 781.

was durch einen erhöhten Recall-Wert dargestellt wird.[217] *Weiss, Indurkhya und Zhang* benutzen dazu ein sehr gutes Beispiel mit Spam-E-Mails. Wenn ein Spamfilter eine hohe Genauigkeit, jedoch einen geringen Recall-Wert, aufweist, dann bleibt meist Spam im Posteingang. Auf der anderen Seite sind die Mails, die in den Spam-Ordner verschoben werden, aufgrund der hohen Genauigkeit, meistens korrekt als Spam klassifiziert. *Weiss, Indurkhya und Zhang* nennen dies *precision-recall tradeoff*.[218] Erhöht man nämlich beispielsweise die Precision dann sinkt der Recall-Wert und damit steigt die Zahl, der nicht richtig erfassten Dokumente. Dieser Tradeoff kann meistens bei den Klassifikatoren verändert werden, um ihn an die geplante Anwendung entsprechend anzupassen. Die Metriken und Evaluation der Performance helfen dabei, den Algorithmus so anzupassen, wie es für die Anwendung optimal ist.

[217] Vgl. Weiss, Indurkhya und Zhang, 2015, S. 70.
[218] Vgl. Weiss, Indurkhya und Zhang, 2015, S. 71.

Kapitel 5

Praktische Umsetzung am Beispiel Amazon

Nachdem die vorangegangenen Kapitel die theoretischen Hintergründe beleuchtet haben, folgt in diesem Kapitel die Anwendung anhand eines praktischen Beispiels. Im Beispiel soll dazu der Onlineversandhändler Amazon als Unternehmen dienen, um aufzuzeigen, wie die Sentiment Analyse von Kurztexten im Unternehmenskontext genutzt werden kann.

5.1 Zielsetzung

Das Ziel dieses praktischen Beispiels ist es, aufzuzeigen, wie Sentiment Analyse im Unternehmenskontext bei Amazon genutzt werden könnte. Dabei liegt der Fokus auf dem Kundendienst. Es sollen Twitter-Daten genutzt werden und mit Hilfe eines lexikonbasierten Ansatzes analysiert werden. Neben der eigentlichen Sentiment Analyse soll auch das Sammeln der Daten, die Verarbeitung und Analyse, sowie eine Visualisierung der Ergebnisse mit Hilfe eines Dashboards, in diesem Beispiel behandelt werden.

5.2 Amazon

Bei Amazon handelt es sich um einen der größten Onlineversandhändler weltweit, mit einem Jahresumsatz von circa 136 Milliarden Dollar[219]. Amazon bietet dabei in den USA circa 280 Millionen und in Deutschland rund 150 Millionen Produkte in ihrem Onlineshop an (Stand 2014)[220]. Neben den eigentlichen Produkten im Onlineshop ist die Produktpalette von Amazon

[219] Manney, 2017.
[220] Jordan, 2014.

durch weitere Dienste, die das Unternehmen darüber hinaus anbietet, jedoch noch breiter aufgestellt. Dazu gehört unter anderem das Abonnement-Modell Amazon Prime, bei dem Kunden gegen eine Gebühr, zum Beispiel schnelleren Versand, Video-Streaming oder Zugriff auf eine Leihbücherei für E-Books, erhalten. Entsprechend der Größe erhält Amazon natürlich auch eine große Menge an Supportanfragen jeglicher Art. Eine Quelle dabei ist der Kurznachrichtendienst Twitter. Hier hat Amazon neben landesspezifischen Accounts wie @AmazonDE oder dienstspezifischen Accounts wie @AmazonVideo auch einen eigenen Account (@AmazonHelp[221]) lediglich für Supportanfragen. Dort werden an 24 Stunden und sieben Tagen die Woche Anfragen beantwortet. Der Account kann in sieben verschiedenen Sprachen kontaktiert werden. Darüber erhalten die Nutzer meist vergleichsweise schnell eine Antwort zu ihrem Problem. Auf Nachrichten, die an diesen Account gerichtet werden, liegt der Schwerpunkt, dieser im Folgenden praktischen Umsetzung.

5.3 Social Media

Soziale Netzwerke, beziehungsweise Social Media, ist heutzutage kaum mehr aus dem Alltag der Menschen, vor allem der Jüngeren, wegzudenken. Viele, gerade in der jüngeren Generation, sind meist in mehreren sozialen Netzwerken gleichzeitig aktiv, wie Facebook, Twitter oder Instagram. Dabei dienen sie meist dazu, persönliche Inhalte zu teilen und mit anderen Personen zu kommunizieren. Neben Privatpersonen nutzen natürlich auch Unternehmen soziale Medien, um unter anderem Werbung zu machen oder mit dem Kunden zu interagieren. Jede größere Firma hat heute beispielsweise eine Seite im sozialen Netzwerk Facebook oder einen Twitter-Account. Die Basis des sozialen Netzwerks bilden dabei jedoch die Verbindungen zwischen individuellen Nutzerprofilen[222]. Die folgenden zwei Abschnitte sollen einerseits das soziale Netzwerk Twitter, als auch den Nutzen und die Herausforderungen für Unternehmen, in sozialen Netzwerken, näher beschreiben.

5.3.1 Twitter

Bei Twitter handelt es sich ebenfalls um ein soziales Netzwerk. Genau spezifiziert, ist es jedoch ein sogenannter Microbloggingdienst. Dabei sind im Vergleich zu einem regulären Blog die Länge eines Posts künstlich auf 140

[221] `https://twitter.com/AmazonHelp`
[222] Vgl. Schmidt, 2013, S. 12.

Zeichen begrenzt[223]. Ein Post wird bei Twitter *Tweet* genannt. Der Name Twitter, beziehungsweise der Tweet, stammt von dem englischen Verb to twitter / to tweet ab, was übersetzt, zwitschern bedeutet. Twitter wurde dazu entwickelt anderen mitzuteilen, was man gerade tut und zu lesen, was andere gerade tun. Doch vor allem Unternehmen nutzen Twitter um beispielsweise Angebote oder Neuigkeiten zu Produkten zu verbreiten. Ein Tweet kann öffentlich zugänglich gepostet werden oder privat an einzelne Personen. Möchte man einen Tweet an eine bestimmte Person richten, beginnt dieser mit dem @-Zeichen, gefolgt von dem Benutzernamen der Person. Neben der Möglichkeit Tweets an bestimmte Personen zu richten, ist es auch möglich, bestimmten Nutzern zu folgen. Dadurch bekommt man immer die öffentlichen Tweets der Person, der man folgt, in seinem eigenen Feed angezeigt. Gerade bekannte Accounts, wie von Prominenten oder Unternehmen, haben eine hohe Anzahl *Follower*, also Leute, die diesen Account verfolgen. Durch eine hohe Anzahl an Followern erreicht man entsprechend auch eine hohe Reichweite bei Twitter, da man viele Leute auf einmal erreichen kann. Ein weiterer wichtiger Mechanismus bei Twitter sind sogenannte *Retweets*. Bei einem Retweet handelt es sich um einen Tweet einer Person, der von einer anderen Person kopiert, beziehungsweise geteilt, wurde. Hat der ursprüngliche Verfasser mit dem Tweet nur seine eigenen Follower erreicht, so wird die Reichweite durch einen Retweet erhöht, da die Follower des Nutzers, der den ursprünglichen Tweet geteilt hat, die Nachricht dann ebenfalls sehen[224]. Durch Twitter sind auch die sogenannten *Hashtags* populär geworden. Dabei handelt es sich um ein Wort, dem das Raute-Zeichen vorangestellt ist. Der Hashtag ist eine Art Kennwort, um anzuzeigen, dass der Tweet zu einer Gruppe anderer Tweets mit dem selben Thema gehört.

5.3.2 Nutzen und Herausforderungen für Unternehmen

Für Firmen bieten soziale Netzwerke enorme Möglichkeiten, da Privatpersonen und damit Kunden, beziehungsweise potenzielle Kunden, dort meist viel über sich preisgeben. Auch die Daten, die der Anbieter des sozialen Netzwerks, dort sammelt werden unter anderem für auf den Benutzer zugeschnittene Werbung genutzt. Die Informationen aus den Netzwerken können sehr wertvoll sein und, beispielsweise Unternehmen, dabei helfen, zu verstehen, ob ihr Produkt gut oder schlecht von den Kunden aufgenommen wird. Anhand von Twitter-Trends können Unternehmen auch sehen,

[223] Vgl. Barczok, 2008, S. 1; Vgl. Zarrella, 2010, S. 39.
[224] Vgl. Zarrella, 2010, S. 49.

was die Nutzer gerade beschäftigt. Bei den Trends analysiert Twitter das Vorkommen von, zum Beispiel Hashtags. Hashtags, die häufig genutzt werden, steigen in der Trendliste nach oben auf.

Natürlich kann der Nutzen der Informationen, die aus den Daten der Nutzer gewonnen werden, für Unternehmen sehr groß sein. Auf der anderen Seite stellt diese Art der Quelle jedoch auch viele Schwierigkeiten und Herausforderungen für Unternehmen dar. Dazu gehört unter anderem eine Sicht auf diese Daten, die *Richard Heimann* im Buch *Social Media Mining with R* teilt. Er sieht die Gefahr darin, dass Unternehmen Twitter als einfach verfügbare und wertvolle Datenquelle ansehen, die die Masse der Menschen wiedergibt[225]. Natürlich sind die Daten über Schnittstellen einfach abzurufen und ebenso können die Daten wertvolle Informationen enthalten fraglich ist jedoch natürlich immer, ob die Daten auch die Masse der Menschen darstellen, da die Hauptnutzer meist jung sind, aus städtischen Gegenden kommen und der Mittelschicht angehören[226]. Damit würden die Daten nicht der Masse entsprechen. Ebenso ist es aber natürlich auch immer abhängig von der Anwendung und den Erkenntnissen, die man aus den Daten schöpfen möchte. Für bestimmte Zielgruppen, die man analysieren möchte, kann die Quelle ideal sein.

Auch scheint gerade bei Daten aus Twitter, durch die Beschränkung auf 140 Zeichen, die Aussagekraft der Tweets stark limitiert. Doch aufgrund der Beschränkung drücken sich die Nutzer meist präziser und klarer mit ihrer Aussage aus. Die präzise und klare Formulierung macht es zum Beispiel bei der Sentiment Analyse einfacher, als bei langen und ausschweifenden Texten. Bei Analysen, die auf Machine Learning-Ansätze bauen, kann ein Problem fehlende Trainingsdaten sein, da diese in der Regel einfach nicht vorhanden sind. Dies bedeutet entweder, großen manuellen Aufwand um zum Beispiel ein Trainingsset zu erstellen, oder dass man auf Ansätze setzen muss, wo kein Training nötig ist[227].

Darüber hinaus kann die sogenannte *Noise* Analyseergebnisse verfälschen. Bei Noise handelt es sich um Daten, die nutzlos sind und nicht interpretiert werden können. Dies kann zum Beispiel ein Tweet sein, der aus einer

[225] Vgl. Richard Heimann, 2014, S. 44.
[226] Vgl. Richard Heimann, 2014, S. 44.
[227] Vgl. Garg und Kumar, 2015, S. 6.

sinnlosen Aneinanderreihung von Buchstaben besteht und damit auch keine Aussage besitzt. *Zafarani, M. A. Abbasi und H. Liu* nennen das Problem *Noise Removal Fallacy*[228]. Das Problem mit Noise auf der einen Seite ist, dass bei dem Versuch es aus dem Datenset zu entfernen, wichtige Informationen verloren gehen können. Auf der anderen Seite kann Noise aber auch so komplex sein, dass es die Analyse so stark beeinflusst, dass diese nicht mehr aussagekräftig ist. Aber auch bei aussagekräftigen Tweets kann die Qualität der Daten schlecht sein oder der Nutzer hat in seinem Tweet viele Abkürzungen benutzt[229]. Gängige Abkürzungen müssen dann in der Analyse abgefangen werden, um trotzdem gute Ergebnisse liefern zu können. Dies zeigt, dass die Analyse von Social Media Daten und vor allem von Twitter, gewisse Schwierigkeiten aufweisen kann.

5.3.3 Twitter API

Ähnlich wie andere soziale Netzwerke stellt Twitter Schnittstellen, sogenannte Application Programming Interfaces (kurz: API), bereit, um auf die Daten des Dienstes zugreifen zu können. Zwei der Schnittstellen sollen im Folgenden kurz betrachtet werden, die *REST-API* (Representational State Transfer) und die *Streaming-API*. Neben diesen gibt es unter anderem auch eine API um Twitters Werbeplattform zu nutzen, die in diesem Falle aber nicht relevant ist.

Um auf eine der Schnittstellen zugreifen zu können, muss zuerst eine Anwendung bei Twitter erstellt werden. Dazu wird lediglich ein Twitter-Account benötigt, mit dem die Anwendung einfach bei Twitter im Developer-Bereich angelegt werden kann. Dies bedeutet, dass man eine Anwendung anlegt und man diese damit autorisiert, auf den eigenen Account zuzugreifen. Dabei wird verhindert, dass mit einer Benutzername-Passwortkombination auf die API zugegriffen werden kann. Die Autorisierung wird dabei mit Hilfe des OAuth-Protokolls (Open Authorization) durchgeführt. Das Protokoll ermöglicht es, auf eine API zugreifen zu können, ohne dass der Nutzer das eigene Passwort oder den eigenen Benutzernamen benutzen muss. Es werden dabei von der Anwendung, die die API bereitstellt, Access Tokens generiert, mit denen dann der Zugriff auf die API erfolgen kann[230] Im Bereich von Web-Anwendungen und gerade bei Social Media-Anwendungen

[228] Vgl. Zafarani, M. A. Abbasi und H. Liu, 2014, S. 3.
[229] Vgl. Aggarwal und Zhai, 2012, S. 392.
[230] Vgl. Boyd, 2016, S. 10.

ist OAuth der defacto Standard[231].

Twitter stellt über seine API vier verschiedene Objekte zur Verfügung: Tweets, Benutzer, Entitäten und Orte[232]. Die APIs geben dann die Ergebnisse im JSON-Format an den Nutzer zurück. Bei dem Tweets-Objekt handelt es sich um die Tweets, auf denen die gesamte Kommunikation auf Twitter basiert. Es wird beim Tweet-Objekt jedoch nicht nur der Text, den der Tweet enthält, zurückgegeben, sondern viele weitere interessante Metadaten. Neben dem Text kann natürlich auch die Zeit, wann der Tweet erstellt wurde und auch von wem, abgerufen werden. Darüber hinaus gibt es aber noch weitere Metadaten, wie beispielsweise die Sprache, wenn diese erkennbar ist oder ob der Tweet von anderen Nutzern geteilt wurde, das sogenannte Retweeten. Zusätzlich werden auch die anderen drei Objekte im Tweet-Objekt mit ausgegeben. Bei dem Benutzer-Objekt handelt es sich vorrangig um Stammdaten des Benutzers, der den Tweet erstellt hat, dazu zählt beispielsweise auch, welche anderen Benutzer er abonniert hat und von wie vielen er selbst abonniert wurde. Das Entitäten-Objekt umfasst Daten, die Twitter aus dem Tweet herausgelesen hat, wie beispielsweise sogenannte User Mentions, also Nutzer, die man in seinem Tweet nennt, mit Hilfe des @-Symbols. Ebenso jedoch auch Hashtags, die mit dem Rautesymbol beginnen oder URLs. Häufig werden in Tweets verkürzte URLs angegeben. Es wird im Entitäten-Objekt sowohl die verkürzte als auch die ursprüngliche, nicht verkürzte URL, mit ausgegeben. Mit dem Orte-Objekt werden Metadaten zum Standort des Verfassers ausgegeben, wie unter anderem das Land und die Stadt, in der sich der Benutzer befindet.

Bei der REST-API handelt es sich um eine Schnittstelle, in die auch eine sogenannte Search-API integriert ist, wodurch sich komplexe Filtermöglichkeiten ergeben, wenn man über die REST-API beispielsweise Tweets abrufen möchte[233]. Eine Suche nach bestimmten Hashtags oder Nutzern wird dadurch vergleichsweise einfach ermöglicht. Die Daten können mit HTTP-GET-Befehlen abgerufen werden. Es können jedoch nur Daten aus den letzten sieben Tagen abgerufen werden und dabei handelt es sich immer nur um einen Teil der Daten. Twitter beschreibt die Search-API entsprechend so: „the Search API is focused on relevance and not completeness“[234]. Man kann also nicht davon ausgehen, dass man alle Daten bekommt, dafür aber

[231] Vgl. Russell, 2013, S. 13.
[232] Vgl. Twitter, 2017a.
[233] Vgl. API, 2017.
[234] API, 2017.

die Relevanten. Zusätzlich dazu ist auch der Zugriff auf die API limitiert. Das Limit liegt hier bei 15 Anfragen pro 15 Minuten.

Die Streaming-API ist mehr dazu geeignet in Echtzeit die Twitter-Daten abzurufen. Der Unterschied liegt schon im grundsätzlichen Aufbau der API. Wurde bei der REST-API für eine Anfrage auch nur eine Antwort generiert und zurückgeliefert und entsprechend die Verbindung nach der Antwort terminiert, so wird bei der Streaming-API eine dauerhafte Verbindung zur API aufgebaut[235]. Die Anwendung baut diese Verbindung mit bestimmten Parametern auf, um bestimmte Tweets zu filtern. Dadurch werden dann passende Tweets von Twitter über die API zur Anwendung gepusht. Die Filtermöglichkeiten sind ähnlich divers wie bei der REST-API, jedoch ist man bei bestimmten Filterkriterien eingeschränkt, da diese nicht wie bei der REST-API und-verknüpft, sondern oder-verknüpft sind.

5.4 Entwurf und Umsetzung

Dieser Abschnitt behandelt den Entwurf zur Analyse der Tweets und der darunterliegenden Architektur. Die Architektur und der damit einhergehende Versuchsaufbau, sowie genutzte Applikationen, sollen im Abschnitt 5.4.1 näher erläutert werden. Im Anschluss daran wird die eigentliche Umsetzung behandelt und dargestellt.

5.4.1 Architektur

Die Architektur, in Abbildung 5.1 zu sehen, kann dabei in verschiedene Bereiche unterteilt werden. Kurz erläutert, wird im Bereich der Datenquellen auf lediglich zwei Quellen zugegriffen, einmal Twitter, beziehungsweise die Twitter Streaming API und das Sentiment-Lexikon als Textdatei. Die Infrastruktur für die Verarbeitungs- und Analyseebene baut auf einer Amazon Web Services (AWS) Elastic-Compute-Cloud-Instanz (auch EC2-Instanz genannt) auf. Auf der Instanz ist als Hadoop-Plattform Cloudera mit HDFS als Dateisystem, Flume zur Aufnahme der Twitterdaten ins HDFS, sowie Hive zur Verarbeitung und Analyse der Daten, installiert. Mit Hilfe von QlikSense kann dann auf die Plattform beziehungsweise direkt auf Hive, zugegriffen werden, um die Daten abzurufen und zu visualisieren.

[235] Vgl. Twitter, 2017b.

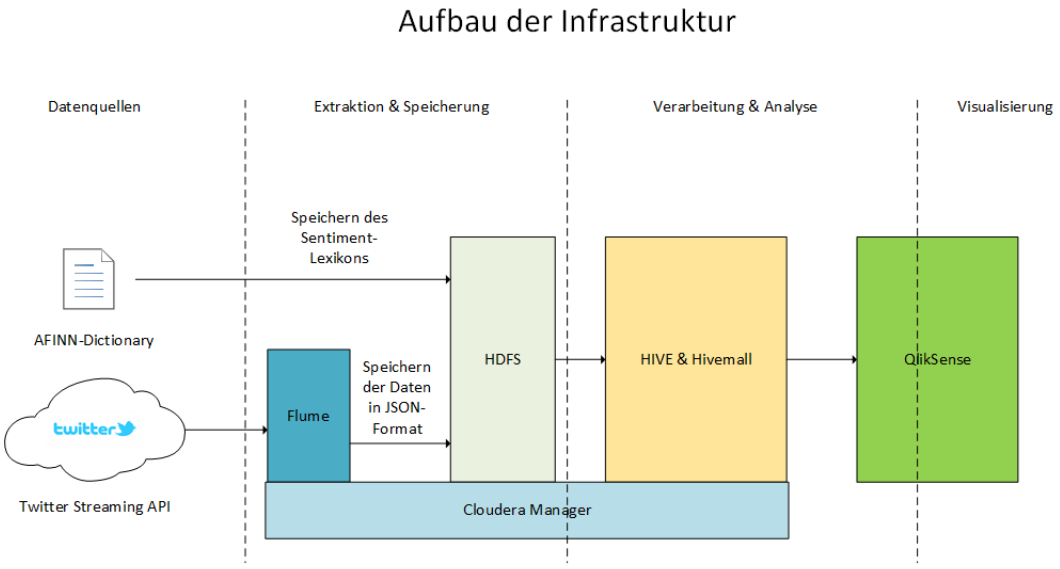

ABBILDUNG 5.1: Architektur des Versuchaufbaus

Bei den Datenquellen handelt es sich sowohl um strukturierte als auch unstrukturierte, beziehungsweise semi-strukturierte, Daten. Das Sentiment-Lexikon hat dabei eine fest vorgegebene Struktur und kann als strukturiert angesehen werden. Bei den Tweets, die über die Streaming API gesammelt werden, handelt es sich jedoch um semi-strukturierte Daten. Die Tweets werden von der API in einem JSON-Format ausgegeben. Innerhalb dieser Struktur sind die Daten aber unstrukturiert, deswegen kann hier von semi-strukturierten Daten gesprochen werden. Die eigentliche Speicherung, Verarbeitung und Analyse der Daten wird mit der bereits erwähnten Hadoop-Distribution von Cloudera ermöglicht. Dazu soll Cloudera auf einer AWS EC2-Instanz installiert und entsprechend der nötigen Dienste, wie Flume, HDFS und Hive, konfiguriert werden. Der Vorteil bei der Installation in der Cloud ist die schnelle und einfache Verfügbarkeit von Hardware und nötiger Leistung. Mit Hilfe von Flume wird auf die Twitter API zugegriffen und die Tweets abgerufen, welche dann persistent in HDFS im JSON-Format abgelegt werden. Neben den Twitterdaten wird auch das Sentiment-Lexikon im HDFS abgelegt. Mit Hilfe von Hive und Hivemall[236], einer Erweiterung für Hive um Machine-Learning-Funktionen, können die Daten im JSON-Format in Tabellenform gebracht und verarbeitet und analysiert werden. Zur Analyse wird dann auch das Sentiment Lexikon in Hive in eine Tabelle geladen. Nach der Analyse kann mit Hilfe eines speziellen Hive-Connectors

[236] `https://hivemall.incubator.apache.org/`

von QlikSense direkt auf Hive zugegriffen und Daten aus den Tabellen abgerufen werden. Mit Hilfe von QlikSense, einem Tool zur Datenvisualisierung, sollen dann die Ergebnisse aus der Verarbeitung und Analyse der Daten visualisiert und ansprechend dargestellt werden. Im Abschnitt 5.4.2 werden die einzelnen Schritte noch im Detail erläutert.

5.4.2 Umsetzung

Um den in Abschnitt 5.4 skizzierten Entwurf umzusetzen, muss im ersten Schritt die Infrastruktur aufgesetzt werden. Dazu wird in diesem Beispiel eine Instanz in Amazons EC2 genutzt. Es handelt sich um eine x2.large - Instanz, um genügend Rechenleistung zur Verfügung zu stellen. Bei der Hadoop-Distribution handelt es sich um Cloudera, in der Version CDH 5.10. Dabei wurde das Standard-Setup an Komponenten von Cloudera mitinstalliert, unter anderem Flume, Hive, Hue und HDFS. Cloudera ist eine von vielen Hadoop-Distributionen. Die Distributionen unterscheiden sich häufig durch die von Unternehmen entwickelten zusätzlichen Funktionalitäten. Bei Hue handelt es sich um eine Weboberfläche, zur Nutzung von beispielsweise Hive- oder Pigabfragen. Diese kommt im Bereich der Verarbeitung und Analyse der Daten zum Einsatz.

5.4.2.1 Datenquellen

Prinzipiell gibt es bei diesem praktischen Beispiel nur zwei grundlegende Datenquellen. Auf der einen Seite das Textfile, dass das Sentiment-Lexikon enthält und auf der anderen Seite die Tweets, die bei Twitter abgerufen werden. Natürlich ist auch Hive eine Datenquelle für QlikSense zur Visualisierung, der Fokus liegt jedoch auf den zwei grundlegenden Quellen.

Die Schnittstelle von Twitter wurde bereits in Abschnitt 5.3.3 näher erläutert. In der Umsetzung wurde sich gegen die REST- und für die Streaming API entschieden, da Tweets über einen längeren Zeitraum gesammelt werden sollen und diese kontinuierlich in das System fließen sollen, ohne dass immer wieder eine neue Anfrage gesendet werden muss. Die Schnittstelle liefert dabei Daten im JSON-Format zurück. Im JSON-Format sind die Daten in einer Schlüssel-Wert-Beziehung abgelegt. Dabei kann ein Wert jedoch

auch eine Liste weiterer Schlüssel-Wert-Paare sein. Betrachtet man zum Beispiel eine Twitter-JSON-Datei[237] kann dies leicht festgestellt werden. Es entsteht eine Art Baumstruktur. Auf der obersten Ebene befinden sich Daten, wie der Text des Tweets oder ob und an wen der Tweet gerichtet ist. Natürlich bekommt jeder Tweet eine eindeutige Nummer (ID) sowie einen Zeitstempel zugewiesen. In Abschnitt 5.3.3 wurden auch die Objekte der API beschrieben. Der Schlüssel Entitäten in der Datei enthält eine Liste mit weiteren Schlüssel-Wert-Paaren, die zum Beispiel aus dem Tweet extrahierte Benutzernamen oder Hashtags ausweisen. Dies ermöglicht es, auf einfache Weise, eine Analyse anhand von Hashtags durchzuführen. Des Weiteren befindet sich im User-Objekt eine große Anzahl an weiteren Schlüssel-Wert-Paaren. Hier können unter anderem die Sprache des Nutzers, seine Herkunft, Anzahl der Freunde und viele weitere Daten ausgewertet werden. Entscheidend für die Sentiment Analyse ist natürlich eine eindeutige Identifikationsnummer und der Text, also der eigentliche Inhalt des Tweets. Darüber hinaus können die weiteren zur Verfügung stehenden, größtenteils Metadaten jedoch ebenfalls interessante Erkenntnisse erbringen, wenn sie zu den Ergebnissen der Sentiment Analyse hinzugezogen werden.

Neben den Twitterdaten ist natürlich das Sentiment-Lexikon essentiell für die Sentiment Analyse. Dazu wird auf ein bereits bestehendes Sentiment-Lexikon zurückgegriffen. Wie in Abschnitt 4.2.2 beschrieben, ist ein bestehendes Lexikon jedoch meist nur ein Startpunkt. Dies soll im Laufe der Analyse ebenfalls beurteilt werden. Bei dem Lexikon handelt es sich um das AFINN-Dictionary von Finn Årup Nielsen[238], einem dänischen Forscher von der Technical University of Denmark.

Wort	**Sentiment Score**
Wort a	-1
Wort b	2

TABELLE 5.1: Aufbau AFINN-Dictionary

Dabei handelt es sich vereinfacht gesagt, um eine Liste mit englischen Wörtern, denen ein Sentimentwert zwischen minus 5 (sehr negativ) und plus 5 (sehr positiv) zugeordnet wurde. Die Null bedeutet in diesem Falle, dass es sich um ein neutrales Wort handelt. Die Liste umfasst in der aktuellsten Version 2477 Wörter und ist speziell für den Einsatz zur Analyse von Microblogs, wie Twitter, erstellt worden. Dabei wurden die Wörter manuell von

[237] `https://gist.github.com/hrp/900964`
[238] s. Nielsen, 2011.

Nielsen zwischen 2009 und 2011 mit Werten versehen. Dies soll, wie bereits erwähnt, der Ausgangspunkt für die Analyse sein.

5.4.2.2 Extraktion und Speicherung

Damit die Twitterdaten extrahiert werden können, muss auf die Streaming API von Twitter zugegriffen werden. Die Daten werden nach der Extraktion in HDFS abgelegt. Um auf die Streaming API zugreifen zu können, wird in diesem praktischen Beispiel Flume genutzt. Flume ist eine der Komponenten, die Teil des Hadoop-Ökosystems sind, wie bereits in Abschnitt 2.4.2.4 beschrieben, dient es dazu, auf Daten aus verschiedenen Quellen zuzugreifen. Der Datenfluss und die Architektur von Flume ist in Abbildung 5.2 zu sehen.

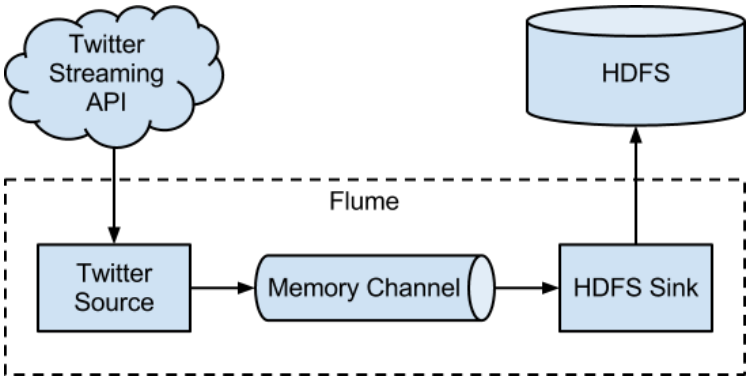

ABBILDUNG 5.2: Flume Prozessablauf Quelle: `http://bit.ly/2pmHC6V`

Auf der linken Seite wird zwischen Twitter und der Source eine Verbindung aufgebaut und Twitter sendet neue Ereignisse, also neue Tweets, an die Source von Flume. Diese legt Flume in einem sogenannten Channel ab. Dabei handelt es sich um eine Art Zwischenspeicher, bevor die Daten über die Sink weggeschrieben werden. Ein Channel kann beispielsweise vom Typ File sein, dann werden die Daten auf dem Dateisystem zwischengespeichert oder vom Typ Memory, dann werden die Daten im Arbeitsspeicher temporär gehalten. Die Sink kann unter anderem HDFS sein, wo die Daten dann persistent abgelegt werden.

Damit Flume die Daten von Twitter abrufen und empfangen kann, muss Flume konfiguriert werden. Als Teil der Konfiguration werden die Source, der Channel und die Sink definiert. Um die Daten von der Twitter Streaming API abrufen zu können, bedarf es einer sogenannten Custom Source. Da Flume die Möglichkeit bietet eine eigene Source zu definieren, können

zahlreiche weitere Quellen angebunden werden, die im Standard nicht vorhanden sind. Die für dieses praktische Beispiel genutzte Konfiguration ist in Anhang A zu sehen. Bei der Konfiguration definiert man immer einen sogenannten Agenten. In der hier genutzten Konfiguration wird dieser einfach *agent* genannt. Zu Beginn der Konfiguration werden die drei Bestandteile von Flume, die Source, der Channel und die Sink bestimmt und mit Namen aufgelistet. Der Channel wird als Memory Channel definiert und die Sink soll HDFS sein.

Im Anschluss daran wird die Source näher definiert. Da es sich um eine Custom Source[239] handelt, wird im ersten Schritt der Typ definiert, in dem das Paket zur Custom Source angegeben wird. Die dazugehörige jar-Datei, die die Custom Source enthält, muss vorher im Pfad */usr/lib/flume/lib/* abgelegt werden, damit Flume darauf zugreifen kann. Im nächsten Scchritt wird die Source an den Memory Channel gebunden, damit die Events der Quelle in den Channel laufen. Die Custom Source ermöglicht es, direkt auf die Twitter Streaming API zuzugreifen. Dazu werden in den nächsten vier Schritten die Keys und Access Token für OAuth zur Authentifizierung gegenüber der API angegeben. Nicht nur der Zugriff auf die API, sondern auch deren Möglichtkeit die Daten zu filtern, kann über die Source genutzt werden. Mit Hilfe von Keywords kann beispielsweise nach Tweets gefiltert werden, die bestimmte Schlüsselwörter enthalten. Im vorliegenden Beispiel sollen Tweets an den Support-Account von Amazon näher betrachtet werden. Als Keyword wird deshalb „@AmazonHelp“ definiert, da so nur Tweets abgegriffen werden, die an AmazonHelp gerichtet sind. Eine der Besonderheiten der Twitter API ist, dass auch ein Kennzeichen für die Sprache des Tweets mit ausgegeben wird, sofern Twitter die Sprache eindeutig identifizieren kann. Dies ermöglicht es gezielt, Tweets nach Sprache auszusuchen. Um das Beispiel zu vereinfachen, wurde sich im Rahmen dieser Ausarbeitung auf englischsprachige Tweets fokussiert. Entsprechend dieser Einschränkung wurde für den Filter Language das Kürzel *en* übergeben, damit nur englischsprachige Tweets gefiltert werden. Weitere Filter wurden an dieser Stelle nicht angewandt.

Im nächsten Schritt wird die Sink näher definiert. Auch die Sink wird an den Memory Channel gebunden und kann dadurch die Events aus dem Channel abrufen und wegschreiben. Die Sink ist vom Typ HDFS, dass bedeutet, dass die Sink die Daten persistent in HDFS speichert. Damit klar ist, wo die

[239] `https://github.com/mmartsen/flume-tools`

Daten gespeichert werden sollen, wird entsprechend der Pfad definiert und in welchem Dateityp, hier DataStream. Man kann darüber hinaus einen Dateiprefix definieren (hier TwitterPublicStream) und einen sogenannten *InUsePrefix*, der angibt, ob eine Datei gerade noch geschrieben wird (hier _). Mit dem Parameter *maxOpenFiles* kann definiert werden, wie viele Dateien die HDFS Sink gleichzeitig offen haben kann. Der Parameter *batchSize* gibt an, wie viele Events pro Batch maximal geschrieben werden können. Die Parameter, die mit *roll* beginnen, können genutzt werden um zu definieren, wann Flume beim Schreiben der Files in HDFS eine neue Datei beginnen soll. In diesem Beispiel wird nach 10.000 Events eine neue Datei begonnen. da der Wert von *rollCount* auf 10.000 eingestellt ist. Mit einem Wert von 0 deaktiviert man den Parameter, wie bei *rollSize* und *rollInterval* zu sehen.

Zuletzt wird noch der Channel näher definiert. Dieser ist vom Typ *memory*, weil es sich um einen Memory Channel handelt und wird mit der Kapazität von 100.000 Events angegeben. Dies bedeutet, dass 100.000 Events im Channel zwischengespeichert werden können. Mit dem Parameter *transactionCapacity* wird dann definiert, wie viele Events pro Transaktion von der Quelle oder zur Sink geholt, beziehungsweise gegeben werden (in diesem Fall 1.000). Mit Hilfe dieser Konfiguration ist es Flume möglich, auf die Twitter API so zuzugreifen, wie es für das Beispiel hier nötig ist. Wird der Dienst dann neugestartet, nutzt Flume automatisch die festgelegte Konfiguration und beginnt Tweets zu speichern. Um eine größere Menge an Tweets zu sammeln, wurde Flume mehrere Tage aktiviert. Die Dateien, die mit TwitterPublicStream anfangen, können dann in HDFS unter */user/Hadoop/twitter_data/* eingesehen werden. Innerhalb der DataStream Dateien sind die Daten textuell im JSON-Format abgelegt.

5.4.2.3 Verarbeitung

Nachdem die Daten via Flume gesammelt wurden, müssen diese weiterverarbeitet werden, da die rohen Streamingdateien so nur schlecht strukturiert analysiert werden können. Hierzu wird Hive, eine ebenfalls in Abschnitt 2.4.2.4 beschriebene Komponente der Hadoop-Plattform, eingesetzt. Dabei handelt es sich um eine Art Data Warehouse für große verteilte Datenmengen, auf die, mit Hilfe von SQL beziehungsweise HiveQL, zugegriffen werden kann. Der Vorteil an Hive ist, dass die Daten nicht zuerst ins Data Warehouse geladen werden müssen, sondern die Struktur kann direkt auf

die Daten im HDFS projeziert werden.

Im vorliegenden Fall sollen Daten im JSON-Format in ein Tabellenformat übertragen werden. Hive ist von Haus aus sehr flexibel im Umgang mit Daten jeglicher Art. Um JSON Daten direkt einzulesen, kann über einen, von Cloudera bereitgestellten, JSON Serializer Deserializer oder auch JSON SerDe genannt, realisiert werden. Dazu muss Hive nur die Jar-Datei bekannt gemacht werden, die den SerDE enthält, wie zu sehen in Anhang B, bei Hinzufügen der Jar-Dateien.

Im Anschluss daran kann die Haupttabelle mit den Twitterdaten in Hive angelegt werden. Der Code zum Anlegen der Haupttabelle ist in Anhang B unter *Anlegen der Tweets-Tabelle* zu finden. Die erste Besonderheit beim Anlegen der Tabelle ist, dass es sich um eine externe Tabelle handelt. Dies bedeutet, dass keine Daten in die Tabelle geladen werden, sondern die Tabelle nur ein Konstrukt von Metadaten ist, das über einen Dateipfad gelegt wird. Hier wird deutlich, wie Hive es ermöglicht eine Struktur auf Daten direkt im HDFS zu projezieren. Am Ende des Befehls wird das Format mit SerDe angegeben und auf den JSON-SerDe von Cloudera verwiesen und zudem der Pfad der Twitterdaten angegeben. Der Befehl legt die Tabelle mit den wichtigsten Daten, die aus den Twitterdaten genutzt werden können, an. In den Rohdaten befinden sich noch weitere Felder, die für die Auswertungen jedoch nicht weiter relevant sind. Der Data Definition Language Befehl (DDL-Befehl) zeigt auch einige Felder, die vom Typ Array sind, zum Beispiel das Feld Entities. Dabei handelt es sich um im JSON-Format tiefer verzweigte Äste. Dies bedeutet, dass das Feld Entities vom Typ Array weitere Felder enthält, ein Beispiel wäre das Feld Hashtags, dass die Hashtags, die im Tweet genutzt wurden, auflistet. Solche Array-Felder werden in der Tweets-Tabelle jedoch nicht aufgelöst angezeigt. Das bedeutet, dass alle Werte, beziehungsweise Felder in diesem Array hintereinander in der Tabelle angezeigt werden. Um auch diese Daten nutzbar zu machen, werden für die drei Array-Felder in der Definition sogenannte Views erstellt.
In Abbildung 5.3 ist zu sehen, wie der Aufbau dann aussieht. Es werden drei Views angelegt, deren Quelle die Haupttabelle *tweets* ist. Der View *tweets_entities* enthält die ID des Tweets und die einzelnen Felder des Arrays *entities*, wie Hashtags oder genannte User im Tweet. Im View *tweets_retweeted* wird das Array *retweeted* aufgelöst und enthält Informationen, ob ein Tweet geteilt wurde, wie häufig, von wem und mit welchem Text. Zu guter letzt

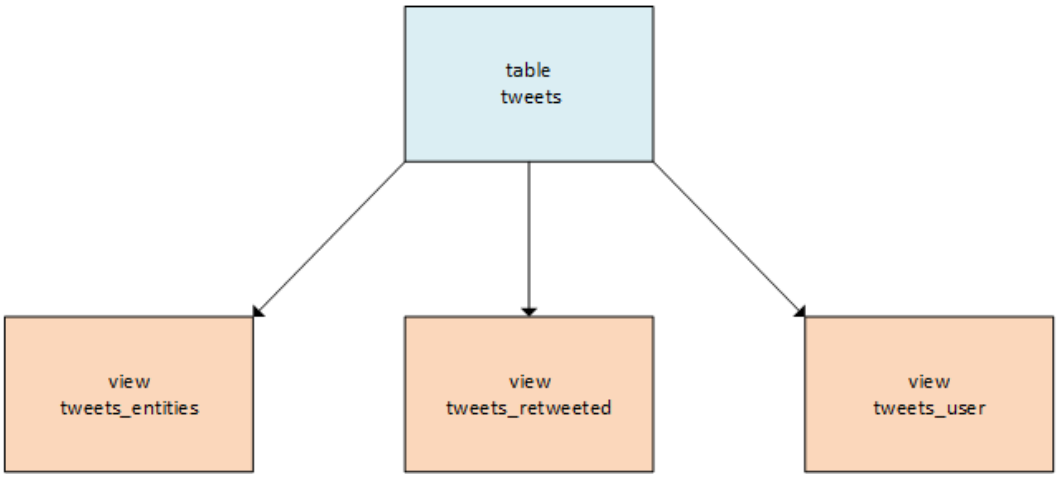

ABBILDUNG 5.3: Hive-Datenmodell zur strukturierten Darstellung der Tweets

wird das Array *user* im View *tweets_user* aufgelöst und enthält viele Informationen über den Verfasser des Tweets, wie der Ort, von dem der Tweet gesendet wurde, wie viele Follower der Nutzer hat und so weiter. Damit wurden die Twitterdaten erfolgreich in Tabellenform überführt und können nun für die Analyse weiterverwendet werden.

5.4.2.4 Analyse

Die Sentiment Analyse soll ebenfalls mit Hilfe von Hive durchgeführt werden. Gerade wenn es um die Vorbereitung der Daten für eine Sentiment Analyse geht, ist Hive von Haus aus eigentlich nicht das Werkzeug der Wahl. Einige Schritte, wie die Tokenization der Texte, kann nur umständlich oder mit schlechter Qualität durchgeführt werden, da ein einfaches splitten der Texte nach Leerzeichen in der Regel nicht ausreicht und Satzzeichen beispielsweise ebenfalls beachtet werden müssen. Darüber hinaus ist es vergleichsweise umständlich Stopwörter zu entfernen, um die Wörter der Tweets auf das Wesentliche zu limitieren.

Um dies gewährleisten zu können, entwickelte Dr. Makoto Yui[240], ein japanischer Forscher, die Machine-Learning-Bibliothek Hivemall für Hive. Hivemall ergänzt Hive um Funktionen aus dem Bereich des Machine-Learnings. Dabei wird die Bibliothek, ähnlich wie der JSON-SerDe von Cloudera als JAR-Datei eingebunden (siehe Anhang B *Hinzufügen der JAR-Dateien*). Das Paket ist mittlerweile Teil des Apache Inkubators. Dabei handelt es sich um den Einstiegspfad für Projekte, die Teil der Apache Software Foundation

[240] `myui.github.io`

(wie z.B. Hadoop) werden wollen. Hivemall bietet unter anderem Funktionen zur Tokenization, sowie zum Entfernen von Stopwörtern. Nachdem die JAR-Datei eingebunden ist, müssen Hive noch die Hivemall-Funktionen bekanntgemacht werden. Die dazu nötigen Befehle können in Anhang B unter *Anlegen der Hivemall-Funktionen*, eingesehen werden. Über die in Anhang B gezeigten Funktionen hinaus existieren noch weitere Funktionen, die aber für diese Ausarbeitung keine Relevanz haben.

Als Ausgangstabelle für die Analyse gilt, die im Schritt der Verarbeitung angelegte, Tabelle *tweets*. Hier liegen die Daten aus Twitter in roh in Tabellenform ab, was es sehr erschwert diese zu analysieren, beispielhaft in Tabelle 5.2 dargestellt. Anhand der Beispiele in Tabelle 5.2 ist bereits zu sehen, dass die Daten zuerst bereinigt und in Token gespaltet werden müssen, damit eine Analyse mit Hilfe eines Sentiment-Lexikons möglich wird.

tweets.id	tweets.text
1	@AmazonHelp I spent 35 mins on the phone with call centre. Solution not ideal, I won't get the item today. So much...
2	@AmazonHelp Already spoke.. But they are not able to help and they don't have right information
3	@AmazonHelp rubbish packaging by Amazon..... Product was not covered with bubbles or any other cushion...

TABELLE 5.2: Beispieltweets aus der Tabelle *tweets*

Damit die Tweets entsprechend einer Sentiment Analyse unterzogen werden können, wird ein Analyseprozess, wie in Abbildung 5.4 definiert. Dazu wird in einem ersten Schritt mit Hilfe der Hivemall-Funktion *tokenize()* der Tweettext in ein Array von Token umgewandelt. Dabei wird mit in der tokenize-Funktion definierten Trennzeichen, der Text in Token aufgespalten und die einzelnen Token werden in einem Array als Liste abgelegt. Der Tweet Nummer drei, aus der Tabelle 5.2, wird entsprechend als Array ["@amazonhelp", "rubbish", "packaging", "by", "amazon", "product", "was", "not", "covered", "with", "bubbles", "or", "any", "other", "cushion"] ausgegeben. Dies geschieht im View *tokenized* (Anhang B *Anlegen des Views tokenized*).
Aufbauend auf dem View *tokenized* wird dann ein weiterer View erstellt, *tokenized_exploded* (Anhang B *Anlegen des Views tokenized_exploded*). Der Name ist dabei von einer Hive-eigenen Funktion abgeleitet. Durch die Funktion *LATERAL VIEW explode()* ermöglicht es Hive beispielsweise ein Array in einzelne Zeilen aufzubrechen. Dies wird hier genutzt, um das Array mit

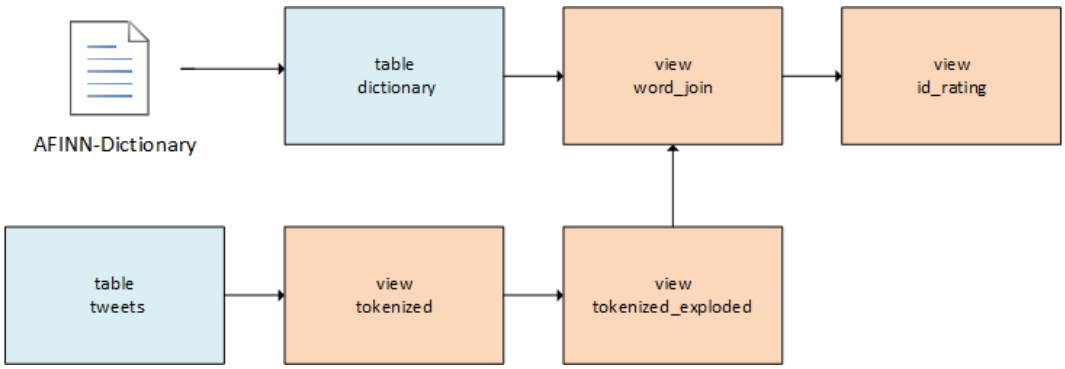

ABBILDUNG 5.4: Analyseprozess in Hive

den Token in eine Spalte *word* zu übertragen, so dass eine Zeile den Schlüssel des Tweets und ein Wort des Tweettextes beinhaltet. Ein Schlüssel kann entsprechend *n* Zeilen mit Wörtern beziehungsweise Token besitzen. Dieser Schritt ist notwendig, um in einem folgenden Schritt die Wörter des Sentiment-Lexikons mit denen der Tweets abgleichen zu können. Eine Besonderheit bei der Erstellung des View *tokenized_exploded* stellt die Funktion *is_stopword()* dar. Dabei handelt es sich wiederum um eine Hivemall-Funktion, die überprüft, ob ein Wort ein Stopwort ist oder nicht. Entsprechend wird bei der Anlage des Views *where not is_stopword(word)* eingesetzt. Dadurch werden nur Wörter in den View übernommen, bei denen es sich nicht um Stopwörter handelt. Im Englischen sind Stopwörter beispielsweise „I, me, myself, we, our, he, him, his“. Nimmt man wieder Tweet Nummer drei, aus dem Beispiel in Tabelle 5.2, würden hier die Wörter „by, was, not, with, or, any, other“ entfernt werden.

Um nun den Wörtern einen Sentimentwert zuordnen zu können, muss zuerst das Sentimentlexikon in Hive angelegt und geladen werden. Dazu wird mit dem DDL-Befehl *Anlegen der Dictionary-Tabelle und Laden des Wörterbuchs* (siehe Anhang B) die Tabelle *dictionary* angelegt und im nächsten Schritt wird das Dictionary aus der Textdatei AFINN.txt in die Tabelle geladen. Im Anschluss können mit dem View *word_join* den Wörtern aus den Tweets Sentimentwerte zugeordnet werden. Daraus resultiert dann eine Tabelle in der folgenden Form:

id	word	rating
1	Wort 1	-1
1	Wort 2	0,5
2	Wort 1	2
2	Wort 2	0,5

TABELLE 5.3: Tabellenform *View word_join*

Nachdem jedem Wort ein Sentimentwert zugeordnet wurde, kann der Gesamtwert für jeden Tweet berechnet werden. Dazu dient im Beispiel der View *id_rating* (siehe Anhang B *Anlegen des Views id_rating*). Dieser View gruppiert pro Tweet und summiert dabei die Sentimentwerte auf. Der daraus resultierende Gesamtwert für jeden Tweet bestimmt, ob ein Tweet positiv, negativ oder neutral klassifiziert wurde. Dort gilt bei einem Wert größer null, dass er als positiv und bei einem Wert kleiner null als negativ anzusehen ist. Bei einem Wert von null ist der Tweet als neutral zu betrachten.

ABBILDUNG 5.5: Beispiele für die Klassifizierung

Bei der Analyse wurden rund 17.000 Tweets klassifiziert. Natürlich muss die Genauigkeit der Analyse noch näher betrachtet werden. Zuerst wird ein View *text_rating* angelegt, der das Rating für jeden Tweet über einen Join auf das Feld id mit dem Text verbindet (siehe Anhang B *Anlegen des Views text_rating*). Anschließend wird von der Gesamtheit aller Tweets eine zufällige Menge von 100 ausgewählt mit Hilfe des Befehls *Zufällige 100 Tweets-Sample* (siehe Anhang B). Um die Genauigkeit zu überprüfen wird jedem Sentimentwert die Sentiment-Klasse zugeordnet (positiv, negativ oder neutral). Anschließend erfolgt eine manuelle Überprüfung der 100 Tweets, um zu prüfen, wie zuverlässig die Analyse ist. Das Ergebnis und die daraus resultierende Genauigkeit kann in Abbildung 5.6 eingesehen werden.
Neben der Genauigkeit, also dem Verhältnis von richtig Klassifizierten zu der Gesamtheit an Klassifizierten, wurden auch die Kennzahlen Precision, Recall, sowie das F1-Maß für alle Klassen berechnet. Daraus resultiert im

		Sentiment Analyse				
		Positiv	**Negativ**	**Neutral**	**Recall**	**F1 score**
Wirklich	**Positiv**	12	0	2	85,7%	52,2%
	Negativ	12	14	12	36,8%	51,9%
	Neutral	8	2	38	79,2%	76,0%
	Precision	37,5%	87,5%	73,1%	64,0%	**Genauigkeit**

ABBILDUNG 5.6: Confusionmatrix mit Qualitätskennzahlen

vorliegenden Beispiel mit einer Genauigkeit von 64%, dass ein Tweet in die richtige Klasse eingeordnet wird. Vor allem bei den positiv Klassifizierten gibt es eine hohe Fehlerrate, was an dem niedrigen Precision-Wert von 37,5% zu erkennen ist. Der hohe Recall-Wert von 85,7% hingegen zeigt, dass wenn ein Tweet wirklich positiv ist, er auch mit einer Wahrscheinlichkeit von 85,7% erkannt und richtig klassifiziert wird. Im Fall der negativ klassifizierten Tweets ist dies genau der umgekehrte Fall. Die Fehlerrate bei der Klassifizierung ist niedrig (Precision-Wert 87,5%), doch ist die Wahrscheinlichkeit, dass ein tatsächlich negativer Tweet auch als negativ klassifiziert wird, nur bei 36,8%. Lediglich die Werte der neutralen Tweets liegen ungefähr im selben Bereich. Das F1-Maß stellt das gewichtete harmonische Mittel zwischen Recall und Precision dar, wobei in diesem Fall beide Werte gleich gewichtet sind. Um die Ergebnisse besser zu veranschaulichen, soll im nächsten Abschnitt, mit Hilfe von Qlik Sense, ein Dashboard entwickelt werden.

5.4.2.5 Visualisierung

Um die Ergebnisse der Analyse besser analysieren und interpretieren zu können, werden in diesem Abschnitt die Daten visualisiert. Zur Visualisierung dient dabei die Anwendung Qlik Sense von Qlik. Qlik Sense wurde gewählt, da es frei nutzbar ist Visualisierungen auf einfache Weise ermöglicht und von Haus aus eine Verbindung zu Hive aufbauen kann. Darüber hinaus ist Qlik Sense über viele Visualisierungserweiterungen, die auf JavaScript basieren, erweiterbar. Um eine Visualisierung der Daten zu ermöglichen, müssen zuerst einmal die Daten in Qlik Sense aus Hive geladen werden. Dazu wird eine Verbindung definiert, wie in Anhang C *Anlegen der Verbindung zu Hive*. Anschließend kann auf die Datenbank und die darin enthaltenen Tabellen zugegriffen werden. Wenn Qlik Sense die Daten lädt werden diese im Arbeitsspeicher gehalten. Dadurch wird ein schnelles Arbeiten mit den Daten ermöglicht. Mit Hilfe des Dashboards soll die Interpretation der

Ergebnisse der Sentiment Analyse vereinfacht werden.

Um dann, über die zuvor erstellte Verbindung zu Hive, die Daten zu laden, muss in Qlik Sense ein Ladeskript erstellt werden. Die Qlik-eigene Skriptsprache ist dabei zum Teil an SQL angelehnt. Das Ladeskript ist in Anhang C (*Qlik Sense Ladeskript*) abgebildet. Der erste Block des Skripts mit den SET-Befehlen wird standardmäßig von Qlik Sense eingefügt. Damit werden Standardvariablen angelegt, um beispielsweise Datums- oder Zahlenformate vorzugeben. Mit dem LIB-Connect-Befehle wird dann die Verbindung zu Hive aufgebaut. Hinter dem Namen *Apache_Hive.amazonaws.com* versteckt sich dann die Verbindungskonfiguration, die zuvor angelegt wurde. Im Anschluss daran wird die erste Tabelle aus Hive *tweets_qlik* geladen. Dabei handelt es sich um die Tweets-Stammdatentabelle, unter anderem mit dem Textinhalt des Tweets, seiner ID und wann er erzeugt wurde. Im darauffolgenden Schritt wird die Tabelle *tweets_qlik*, die sich bereits im Arbeitsspeicher befindet, mit Hilfe des Resident-Befehls erneut geladen. Dieser Befehl dient dazu, die bereits geladene Tabelle erneut zu laden, damit gegebenenfalls weitere Anpassungen erfolgen können. In diesem Fall wird die Tabelle um sogenannte Flag-Felder erweitert. Diese Felder enthalten nur den Wert 1 oder 0. Mit Hilfe der Funktion *SubStringCount()* können Strings innerhalb von anderen Strings gesucht und deren Auftreten gezählt werden. Die Funktion kann dazu genutzt werden, ob ein bestimmter String in einem Tweet auftritt oder nicht. Im vorliegenden Fall sollen damit Objekte identifiziert werden. Mit den Objekten soll eine domänenspezifische Analyse der Daten im Dashboard vereinfacht werden. Kommt in einem Tweet beispielsweise das Wort *delivery* vor, so wird für den Tweet das *Delivery_Flag* auf den Wert *1* gesetzt. Ähnliches wird unter anderem auch für Package, Order, Service und Account durchgeführt. Für das Abomodell Prime von Amazon wurde ebenfalls ein eigenständiges Flag angelegt, damit Anfragen, die speziell von Prime handeln, einfach gefiltert werden können. Die Flags helfen dabei, die Tweets besser zu kategorisieren und für den Mitarbeiter im Kundendienst einen einfacheren Zugriff auf bestimme Tweets zu verschiedenen Objekten zu ermöglichen. Bei der nächsten Tabelle im Ladeskript, die geladen wird, handelt es sich um die Tabelle *qlik_id_rating*, die nur die Tweet-ID und das dazugehörige Rating der Sentiment Analyse enthält. Auch diese Tabelle wird erneut geladen und um das Sentiment (*pos, neg, neutral*) ergänzt. Dafür wird das Rating abgefragt. Bei einem Rating, das kleiner als null ist, wird als Sentiment negativ ausgegeben, größer null als positiv und null als neutral. Diese Tabelle wird dann an die Tweet-Stammtabelle gejoint,

da es sich ohnehin um eine 1:1-Beziehung handelt. Ähnlich wird mit der Tabelle *qlik_tweets_user* verfahren. Diese enthält weitere Daten zum Verfasser des Tweets und wird ebenfalls an die Stammtabelle gejoint. Qlik-Produkte kommen von Haus aus sehr gut mit einzelnen großen Tabellen aus. Große verzweigte Datenmodelle mindern hier in der Regel die Performance, dies ist natürlich immer abhängig vom Anwendungsfall. Zusätzlich wird noch die Tabelle *qlik_words* geladen, diese enthält die ID des Tweets und die einzelnen Wörter, die im Text des Tweets vorkommen. Damit soll eine Wordcloud ermöglicht werden, die die Wörter in Abhängigkeit von der Häufigkeit ihres Auftretens anzeigen soll. Diese wird nicht an die Stammtabelle gejoint, da es sich bei der Beziehung um eine 1:n-Verbindung handelt und über das Feld *id* miteinander verbunden sind. Das daraus entstehende Datenmodell kann in Anhang C (*Qlik Sense Datenmodell*) eingesehen werden.

Das Dashboard (Anhang C *Dashboard - Amazon Twitter Analyse*) selbst besteht dann aus verschiedenen Elementen. Einer der Hauptbestandteile ist die bereits erwähnte Wordcloud, die die Häufigkeit der Wörter darstellt. Die Wordcloud kann auch dabei helfen, sogenannte *Trending Topics* zu identifizieren, dabei handelt es sich um Themen, die gerade häufig erwähnt werden. So kann schnell, auf wichtig werdende Themen, reagiert werden. Bei der Wordcloud handelt es sich um eine Erweiterung, die mit Hilfe von JavaScript erstellt wurde.[241] In der linken oberen Ecke wird die Gesamtzahl aller Tweets ausgegeben und darunter, mit Hilfe eines Kuchendiagramms, die Verteilung der Tweets auf die verschiedenen Sentimente. Unterhalb des Kuchendiagramms wird, mit Hilfe eines Balkendiagramms im Zeitverlauf, die Anzahl der Tweets im Zeitverlauf dargestellt, wodurch ersichtlich ist, wie viele Tweets pro Tag eingegangen sind. Dabei werden die Tage mit mehr als 800 Tweets farblich hervorgehoben. Oberhalb der Wordcloud wird eine Leiste mit den verschiedenen Flags zu Objekten angezeigt. Darüber kann mit einem einzigen Klick, auf Tweets mit einem bestimmten Objekt hin, gefiltert werden. Es können auch mehrere Objekte gleichzeitig ausgewählt werden, sofern alle Objekte innerhalb eines Tweets auftreten. Unterhalb der Wordcloud wird zusätzlich noch eine Tabelle mit den der Tweet-ID, dem Benutzernamen des Verfassers, dem Tweet selbst und auch das dazugehörige Sentiment, sowie das Rating, angezeigt. So haben Kundendienstmitarbeiter mit Hilfe des Dashboards von sehr aggregierten Daten, wie dem

[241] `http://bit.ly/2pjdMhQ`

Kuchendiagramm bis runter auf die ganz detaillierten Daten, also den einzelnen Tweets, einen vollständigen Überblick über die Situation und können entsprechend reagieren. Qlik arbeitet mit einem sogenannten assoziativen Modell, dabei sind alle Objekte im Dashboard miteinander verbunden. Wird beispielsweise auf den Teil der negativen Tweets im Kuchendiagramm geklickt, werden alle anderen Objekte entsprechend auch gefiltert. Ein anderes Beispiel ist, wenn in der Wordcloud auf ein bestimmtes Wort geklickt wird, dann werden alle Objekte so gefiltert, dass nur Tweets mit diesem Wort berücksichtigt werden. Das Filtern funktioniert intuitiv in allen Objekten. Man muss lediglich das Interessante anklicken und schon wird dahingehend gefiltert.

5.5 Einsatz im Unternehmenskontext

Nach der Analyse und der Visualisierung der Daten stellt sich nun die Frage, wie dies im Unternehmenskontext eingesetzt werden kann. Eine Sentiment Analyse für Twitterdaten kann, je nachdem, verhältnismäßig einfach aufgesetzt werden. Der Weg zur erfolgreichen Sentiment Analyse ist jedoch nicht so einfach. Das Aufsetzen einer einfachen Sentiment Analyse reicht in der Regel nicht aus, um für das Unternehmen brauchbare Informationen zu generieren. Das Unternehmen muss sich auf die Analyse verlassen können. In den folgenden Abschnitten soll aus Unternehmenssicht, mit Fokus auf den Twitter-Kundendienst von Amazon, betrachtet werden, wie die Analyse optimiert werden kann, welche Vorteile ein Unternehmen aus der Analyse ziehen kann und welche Schwierigkeiten zu erwarten sind.

5.5.1 Optimierung der Analyse

Eine Genauigkeit von nur 64% sind für den Einsatz der Sentiment Analyse in einem Unternehmenskontext jedoch nicht ausreichend, da die Fehlerrate doch zu hoch ist. Außerdem soll die Annahme bestätigt werden, dass der Großteil der Tweets als negativ einzuordnen ist, da die meisten User den Kundendienst nicht für positives Feedback kontaktieren, sondern um sich, zum Beispiel zu beschweren. In diesem Abschnitt soll die Analyse verbessert werden, um damit die Genauigkeit zu erhöhen. Unter Zuhilfenahme von Qlik Sense, dem Software-Werkzeug zur Datenvisualisierung, das in Abschnitt 5.4.2.5 zur Visualisierung, zum Tragen kommt, kann auf einfache Weise eine nähere Analyse der Tweets durchgeführt werden. Man kann sich

einerseits die Aufteilung auf die verschiedenen Sentimente mit einem Kreisdiagramm veranschaulichen, jedoch viel interessanter ist die sogenannte Wordcloud. Bei der Wordcloud werden die Wörter aus den Tweets, anhand ihrer Häufigkeit, entsprechend größer oder kleiner dargestellt. Mit Hilfe der Wordcloud lässt sich schnell feststellen, in welcher Sentimentklasse, welche Wörter am häufigsten vorkommen.

Betrachtet man die Wordcloud in Anhang C (*Wordcloud vor der Optimierung*) fällt auf, dass viele Token enthalten sind, die für die Sentiment Analyse gar keine Bedeutung haben. Dazu zählt, beispielsweise *@amazonhelp*, *https*, *@amazon*, oder *//t*. Durch diese Token wird die Wordcloud verfälscht, da zum Beispiel der Token *@amazonhelp* in jedem Tweet auftritt und damit entsprechend groß dargestellt wird. Dadurch fallen möglicherweise wichtigere Wörter weniger auf, weil die Größe der Darstellung auf der Häufigkeit basiert und in Relation zu den anderen Wörtern steht. Eine einfache visuelle Analyse wird dadurch erschwert. Die Tabelle, die die Wörter enthält, basiert auf der Tabelle *tokenized_exploded* in Hive. Diese Tabelle enthält die Wörter der Tweets nachdem die Stopwörter entfernt worden sind. Idealerweise würden solche Tokens auch bei dem Entfernen der Stopwörter gleich mitentfernt werden, damit die Visualisierung durch die Wordcloud nicht verfälscht wird. Bei der Analyse der Daten ist auch aufgefallen, dass das Stopwort *not* vorrangig in negativen Tweets auftaucht oder in solchen, die eigentlich negativ sein sollten. Aufgrund der Tatsache, dass es sich um ein Stopwort handelt, wird es jedoch entfernt und bei der Analyse nicht berücksichtigt. Um einerseits die überflüssigen Token in der Wordcloud zu entfernen und das Wort *not* mit in die Sentiment Analyse einfließen zu lassen, muss die Funktion von Hivemall, zum Entfernen von Stopwörtern, entsprechend angepasst werden. Der angepasste Code ist in Anhang B (*Abgeänderte Stopwords-Funktion*) einzusehen. Der Unterschied der Wordcloud vor und nach der Anpassung der Stopwords-Funktion ist sehr deutlich und jetzt werden die wirklich relevanten Wörter viel besser hervorgehoben (siehe Anhang C *Wordcloud nach der Optimierung*).

Mit Hilfe der Wordcloud sollten bei der Betrachtung der Daten, Wörter identifiziert werden, die das Ergebnis verfälschen. Entweder, weil sie einen für diesen Kontext falschen Sentimentwert zugeordnet bekommen haben oder erst gar nicht mit einem Sentimentwert versehen wurden und entsprechend als neutral betrachtet werden. Ein gutes Beispiel für ein Wort, das einen positiven Sentimentwert hat, jedoch in diesem praktischen Beispiel

fast ausschließlich in negativen Tweets auftaucht, ist das Wort „joke“. Im AFINN-Dictionary ist das Wort mit einem Sentimentwert von +2 versehen. Beispiele wie „@AmazonHelp Where is your help, your customer service is big joke“ oder „@AmazonHelp it's becoming a joke ordering through you guys. You have no control over your delivery boys.“ zeigen, dass *joke* meist in einem negativen Zusammenhang genutzt wird. Bei diesem Wort ist dann klar, dass für den Anwendungsfall bei Amazon, der Sentimentwert im Lexikon angepasst werden muss. Im vorliegenden Fall wurde der Sentimentwert von +2 auf -2 geändert.

Ähnlich ist es bei dem Wort „kind“ und trotzdem fällt die Entscheidung hier schwerer, da nicht nur der Sentimentwert abhängig vom Kontext ist, sondern auch die Bedeutung des Wortes eine ganz andere sein kann. Im Sentimentlexikon wird *kind* mit einem Sentimentwert von +2 ausgewiesen. In diesem Fall soll das Wort *kind*, übersetzt soviel bedeuten, wie lieb, freundlich oder nett. Darüber hinaus gibt es aber beispielsweise auch *kind of*, was übersetzt, gewissermaßen bedeutet und keinen Sentimentwert enthält. Um sichergehen zu können in welcher Form das Wort in der Regel auftritt, kann eine weitere Funktion von Hive genutzt werden, *context_ngrams()*. Mit Hilfe dieser Funktion können Wortkombinationen ermittelt werden und in welcher Häufigkeit sie auftreten. Bei dem hier abgebildeten Beispiel werden die 100 häufigsten Bigrams mit *kind* als erstes Wort herausgesucht:

```
SELECT context_ngrams(sentences(lower(text)),
array("kind",null), 100, 1000) FROM tweets;
```

Die Analyse ergibt, dass in rund 90% der Tweets, in denen das Wort *kind* auftritt, nicht die Bedeutung lieb oder freundlich gemeint ist, sondern das Wortpaar *kind of*, und damit keinen Einfluss auf das Sentiment hat. Entsprechend wird das Wort im Zuge der Verbesserung der Analyse aus dem Sentimentlexikon entfernt. Ähnlich wird mit anderen Wörtern verfahren, die gesamte Liste der Änderungen am Sentimentlexikon sind in Anhang D aufgeführt.

Neben Veränderungen der Wörter im Sentimentlexikon sollen, im Zuge der Verbesserung, auch Smileys mit in die Analyse einbezogen werden. Das AFINN-Dictionary bezieht Smileys in seiner Ursprungsform keineswegs mit ein. Lachende und glückliche Smileys sollen positiv bewertet und traurige oder böse Smileys müssen entsprechend negativ bewertet werden. Wie

bereits in Abschnitt 4.2.1 angesprochen, transportieren Smileys die Emotionen in der Regel sehr deutlich und sollten deshalb auch mit in die Analyse einbezogen werden. Dies kann gerade bei bisher neutral bewerteten Tweets, die jedoch Smileys enthalten, den entscheidenden Unterschied machen.

Nach den Anpassungen wird die Sentiment Analyse erneut durchgeführt und die Ergebnisse können verglichen werden. Betrachtet man die Verteilung der Sentimente in Tabelle 5.4, haben sich die Anteile deutlich verschoben. Der Anteil der positiven Tweets ist um rund 11% gesunken, wohingegen der Anteil der negativen Tweets auf 40% gestiegen ist. Der Anteil der neutralen Tweets ist gesunken, um rund 6%.

	Vorher	Nachher
Positiv	33,4%	22,3%
Negativ	22,7%	40%
Neutral	43,6%	37,7%

TABELLE 5.4: Vorher-Nachher-Vergleich der Sentimentverteilung

Die Verteilung nach der Optimierung spricht für die Annahme, dass der Großteil der Tweets als negativ einzuordnen ist. Neben der Verteilung wurde auch anhand einer weiteren Stichprobe von 100 Tweets die Genauigkeit erneut berechnet. Die Genauigkeit konnte im Vergleich um 9,5% erhöht werden. Es konnten alle Werte verbessert werden, bis auf die Precision bei den Neutralen. Besonders hervorzuheben ist der Recall-Wert für die Positiven von 100%, sowie der Precision-Wert für die Negativen von 97,5%. Dies bedeutet, dass die tatsächlich Positiven mit einer Wahrscheinlichkeit von 100% auch als positiv klassifiziert werden. Für die Negativen bedeutet der hohe Precision-Wert, dass die, durch die Analyse klassifizierten, Negativen zu 97,5% korrekt waren.

		Sentiment Analyse				
		Positiv	**Negativ**	**Neutral**	**Recall**	**F1 score**
Wirklich	**Positiv**	12	0	0	100,0%	63,2%
	Negativ	11	39	11	63,9%	77,2%
	Neutral	3	1	21	84,0%	73,7%
	Precision	46,2%	97,5%	65,6%	73,5%	**Genauigkeit**

ABBILDUNG 5.7: Qualitätsmatrix nach der Optimierung

Auch anhand des F1-Score, dem harmonischen Mittel aus Precision und Recall, lässt sich erkennen, dass die Ergebnisse der Analyse verbessert werden konnten. Lediglich der Wert für die neutralen Tweets ist gesunken. Die Analyse der Daten hat jedoch gezeigt, dass die informativsten Tweets die Negativen sind, da die User meist bei einem negativen Anliegen den Kundendienst kontaktieren. Dies wird auch deutlich, wenn man sich die Zeile der wirklich negativen Tweets anschaut. Jeweils elf Tweets sind falsch, als positiv oder neutral, klassifiziert worden. Dies zeigt, dass 61% der Tweets in Wirklichkeit negativ sind. In einem Produktivbetrieb dieser Sentiment Analyse würde man entsprechend versuchen die Anzahl der falsch Klassifizierten weiter zu senken, weil diese auch den Großteil der Tweets ausmachen. Dies stützt zusätzlich die These, dass der Großteil der Tweets negativ einzuordnen ist. Die Verbesserung in diesem Abschnitt hat gezeigt, dass mit einfachen Mitteln die Genauigkeit der Analyse erhöht werden kann und damit auch die Zuverlässigkeit. Letzteres ist gerade im Unternehmensumfeld entscheidend, da man sich auf die Ergebnisse der Analyse verlassen können will. Mit zunehmender Datenmenge kann in einem Unternehmen die Anpassung des Sentimentlexikons regelmäßig wiederholt werden. Eine Revision des Lexikons sollte ein kontinuierlicher Prozess sein, damit man einerseits immer bessere Ergebnisse erzielt und andererseits das Lexikon immer auf die eigenen aktuellen Bedürfnisse abgestimmt wird.

5.5.2 Nutzen für Unternehmen

Ist man in der Lage, eine Sentiment Analyse erfolgreich und mit brauchbaren Ergebnissen durchzuführen, dann stellt sich durchaus die Frage, was ein Unternehmen mit den Ergebnissen machen kann und welchen Nutzen es daraus zieht. Die Sentiment Analyse kann dabei in ganz verschiedenen Bereichen Einfluss nehmen und einen Mehrwert schaffen. Im Anwendungsbeispiel in dieser Arbeit kann dabei die Kundenzufriedenheit ein Bereich sein, die man mit Hilfe der Sentiment Analyse verbessern möchte. *Rapp* stellt in seinem Artikel *Retail Revolution: Die 5 Erfolgsfaktoren* dar, wie wichtig die Kundenzufriedenheit und der Service sind und nennt es einen der Erfolgsfaktoren von erfolgreichen Retail-Unternehmen heutzutage.[242] Die Sentiment Analyse kann dabei helfen, diese weiter zu verbessern, da die Daten schon aufbereitet werden und die Erkenntnisse einfacher identifiziert und in Aktionen umgesetzt werden können. Mit Hilfe der Klassifizierung kann schnell unterschieden werden, welche Tweets von welchen Kunden

[242] Vgl. Rapp, 2016.

eine höhere Priorität haben. Vor allem der Anteil der positiven Tweets bedarf in der Regel keiner priorisierten Bearbeitung, da bei einem Tweet, wie beispielsweise „Shout out to @amazon for delivering my stuff at super lightening speed AND – on a Sunday!!!Kudos @AmazonHelp You Rock“, auf Seiten Amazons, kein Handlungsbedarf besteht. Es handelt sich um sehr positives Feedback, jedoch nicht um ein Anliegen des Kunden, das möglichst schnell bearbeitet werden muss. Relevante Tweets, wie die Negativen, die den Großteil ausmachen, sollten seitens Amazon jedoch möglichst schnell bearbeitet und beantwortet werden. Bei Tweets wie „@AmazonHelp Guess what didn't get delivered today even after calling Amazon twice? I'm so angry and frustrated.“, der mit einem Sentimentwert von *-5* bewertet wurde, also deutlich negativ, besteht dringender Handlungsbedarf. Der Kunde ist verärgert und frustriert, da offensichtlich der Liefertermin nicht eingehalten wurde. Daraus können sich verschiedene Aktionen ableiten lassen:

- Kunde kontaktieren, um Bestelldetails zu erfahren
- Tracking der Bestellung
- Gegebenenfalls Kontakt mit dem Lieferdienst aufnehmen
- Dem Kunden einen neuen Liefertermin mitteilen

Der Kunde ist zufriedener, wenn sich schnell und unkompliziert um sein Anliegen gekümmert wird. Die Klassifizierung, Filter- und Sortiermöglichkeiten im Dashboard helfen dabei, dringende Fälle einfacher zu identifizieren. Es kann beispielsweise auf negative Tweets hin gefiltert werden und vom höchsten negativen Wert absteigend sortiert werden. Dadurch können die negativsten und gegebenenfalls dringendsten Fälle zuerst bearbeitet werden. Natürlich erhöht die Representation der Daten im Dashboard nicht nur die Kundenzufriedenheit auf der einen Seite, sondern auch die Durchlaufgeschwindigkeit des Kundenservice-Prozesses. Der Mitarbeiter muss nicht selber abwägen, ob ein Tweet positiv, negativ oder neutral ist, sondern kann auf die bereits gekennzeichneten Daten zugreifen und mühelos filtern. Geht man davon aus, dass ein Mitarbeiter normalerweise rund 20 Sekunden bräuchte, um einen Tweet zu lesen und zu kategorisieren, Amazon jedoch alleine an englischsprachigen Tweets im Schnitt circa 2000 Tweets pro Tag empfängt, dann würde daraus eine Arbeitsbelastung von circa 11 Stunden pro Tag anfallen. Durch die Sentiment Analyse und das Dashboard können eingehende Tweets, nahe Echtzeit, analysiert und bearbeitet werden.

Der Servicemitarbeiter muss sich nicht mehr um die eigentliche Klassifikation und die damit einhergehende Priorisierung kümmern, sondern kann seine Arbeitszeit dem Kunden und seinem Anliegen widmen. Eine mögliche Erweiterung der Datenanalyse wäre beispielsweise die automatische Extraktion der Bestellnummer des Kunden, falls er diese direkt im Tweet mit angibt. Würde man dann die Bestelldaten mit hinzuziehen, dann könnten zum Tweet direkt die Bestelldaten eingesehen werden. Dies würde eine noch einfachere Bearbeitung des Falls ermöglichen. Da dem Autor kein Zugriff auf Amazon-interne Daten möglich ist, wurde diese Möglichkeit im Anwendungsbeispiel auch nicht mit abgebildet. Die Vereinfachung des Prozesses hilft dabei, dass die Mitarbeiter sich auf die eigentliche Arbeit fokussieren können und damit effizienter sind. Durch die gesteigerte Effizienz können zusätzliche Arbeitskräfte, die sonst zur Bearbeitung nötig wären, gespart und die Kosten im Kundenservice damit gesenkt werden. Die erhöhte Kundenzufriedenheit, bedingt durch einen guten Kundendienst, hilft dabei Kunden langfristig zu binden, da sie sich auf den Kundendienst verlassen können und so auch im Problemfall, der immer auftreten kann, aufgehoben fühlen und das Problem schnellstmöglich gelöst werden kann.

Ein weiterer Punkt, der auch Grundlage für die Kundenzufriedenheit ist und durch die Sentiment Analyse verbessert wird, ist die Qualitätssicherung. Mit Hilfe der automatisierten Klassifikation der Tweets können menschliche Fehler vermieden werden. In der Regel ist auch bei einem vollautomatisierten Klassifikationsprozess keine 100%ige Genauigkeit die Realität. Trotzdem muss der Mitarbeiter, die beispielsweise negativen Tweets lesen und entsprechend darauf reagieren. Auch hier bieten sich zahlreiche Erweiterungsmöglichkeiten des hier gezeigten Anwendungsfalles. Beispielsweise könnten die Objekte, auf die sich der Tweet bezieht, ähnlich wie die Objektfilter (z.B. delivery oder account), extrahiert und entsprechend standardisierte Schritte, zur Problembehandlung, eingeblendet werden. Dies ermöglicht eine Art Prozessstandardisierung und Verkürzung der Durchlaufzeiten, was dann wiederum positiv vom Kunden aufgenommen wird.

Ebenfalls denkbar, wäre eine länderspezifische Analyse der Tweets. Dies ist jedoch nicht ganz trivial, da der Twitternutzer die Einstellung, dass eine Ortsangabe mit übermittelt wird, explizit aktivieren muss. Im Rahmen dieser Arbeit wurden keine geographischen Daten mitanalysiert, da ein großer Teil der gesammelten Tweets keine Geo-Daten enthielt. Die Datenqualität des Location-Feldes ist auch eher suboptimal gewesen. Aus diesem Grund

wurde auf eine Auswertung im Rahmen dieser Arbeit verzichtet. Wäre eine bessere Datenqualität gegeben, könnten länderspezifische Analysen durchgeführt werden, um beispielsweise regionsspezifische Probleme zu identifizieren. So kann zum Beispiel der Lieferdienst, mit dem Amazon in Indien kooperiert, keinen guten Service bieten und die Ware regelmäßig zu spät lieferns. Anhand der Daten könnte schnell erkannt werden wie viele negative Tweets es aus Indien, zum Thema Lieferung, gibt. Daraus kann dann die Erkenntnis gezogen werden, dass die Lieferung in Indien verbessert werden muss. Entweder, in dem man versucht es mit dem bisherigen Lieferdienst zu verbessern, oder einen anderen Lieferdienst in Betracht zieht. Außerdem könnten die Daten, anhand der Geo-Daten, entsprechenden Kundendienstmitarbeitern zugeordnet werden, sodass die Tweets aus Indien auch von dem indischen Kundendienst bearbeitet werden. Dies vereinfacht auch aus kultureller Sicht den Kundenservice, da die Mitarbeiter vor Ort entsprechende Kulturkenntnis haben und mit den lokalen Gepflogenheiten vertraut sind.

Gerade im Social-Media-Umfeld, hier explizit Twitter, kann die Sentiment Analyse auch nicht nur auf Tweets an den eigenen Firmenaccount angewendet werden. Aufgrund der Tatsache, dass die Daten öffentlich zugänglich sind, wäre auch eine Analyse des Wettbewerbs und deren Kundenservice ein möglicher Anwendungsfall. Daraus könnte man verschiedene Informationen generieren, zum Beispiel was den Kunden wichtig ist, wie der Service bei der Konkurrenz aussieht und vergleichbare Fragestellungen. Dies zeigt, wie vielseitig einsetzbar eine Sentiment Analyse im Unternehmenskontext sein kann. Diese Punkte waren jetzt lediglich aus Sicht des Kundenservices via Twitter bezogen. Natürlich können auch klassisch Stimmungen gegenüber Produkten oder Marken, mit Hilfe der Sentiment Analyse, ermittelt und analysiert werden. Im Unternehmenskontext kann die Sentiment Analyse also sehr wertvoll sein und dabei helfen, Prozesse zu optimieren und den Kunden letztendlich zufriedener zu stellen. Wie das Beispiel gezeigt hat, kann mit vergleichbar geringem Aufwand aus Kostensicht, eine Umgebung zur Analyse der Daten aufgebaut werden. Gerade für Unternehmen, die viel mit dem Endkunden in Berührung kommen, lohnt sich eine solche Analyse, da hier auch meist der Kundendienst direkt mit dem Verbraucher in Kontakt steht. Wie das Beispiel zeigt, wird die Möglichkeit Kontakt mit dem Kundendienst über soziale Medien aufzunehmen, schon rege genutzt. Dies macht es umso wichtiger für Unternehmen, diesen Prozess zu vereinfachen.

5.5.3 Schwierigkeiten für Unternehmen

Zunächst muss ein Unternehmen jedoch erst einen Kanal über soziale Medien aufbauen und diesen beim Kunden bekannt machen, damit dieser Weg des Kundendienstes auch genutzt wird. Im Falle Amazons ist dies schon sehr weit fortgeschritten. Neben eigenen Accounts für verschiedene Dienste wurde der AmazonHelp Account geschaffen, um zentral über einen Kanal für verschiedene Sprachen, Support zu leisten. Daraus resultiert auch eine der Schwierigkeiten für Unternehmen. Agiert das Unternehmen global, muss das Unternehmen im Kundendienst auch verschiedene Sprachen verarbeiten können. Einerseits müssen die Mitarbeiter entsprechend die Sprache beherrschen, andererseits möchte man aber auch eine Analyse der Daten durchführen. Dazu muss die Analyse auf die unterschiedlichen Sprachen angepasst werden. Hierbei ist wichtig zu beachten, dass die Analyse nicht verschiedene Sprachen in einem verarbeiten kann. Die Daten müssten nach Sprache getrennt und anschließend separat analysiert werden. Jede Sprache hat ihre eigenen Besonderheiten und diese sollten in der Analyse berücksichtigt werden. Dies kann einen, im Vergleich zu einem rein einsprachigen Kundendienst, deutlich höheren Aufwand bei der Analyse bedeuten.

Auch die Besonderheit von einer Sprache kann schon zu Schwierigkeiten führen. Hier müssen viele Faktoren beachtet werden, die die Analyse der Tweets erschweren können. Dies können unter anderem Abkürzungen sein, da der Kunde versucht, möglichst viel Inhalt in den, auf 140 Zeichen begrenzten, Tweet zu bekommen. Solche Dinge müssen ebenfalls bei der Analyse berücksichtigt und abgefangen werden. Auch Ironie und Sarkasmus bereiten bei der Analyse Schwierigkeiten und können das Ergebnis auf leichte Weise verfälschen. Hier haben Ansätze von Wissenschaftlern, wie in Abschnitt 4.1 gezeigt, dass auch diese Besonderheit mit bestimmtem Aufwand erkannt und entsprechend interpretiert werden kann.

Die größte Schwierigkeit bei der Sentiment Analyse besteht einerseits in der Analyse selbst, aber noch viel mehr in der Interpretation und Weiterverarbeitung der Ergebnisse. Natürlich versucht ein Unternehmen dem Kunden so schnell und so gut wie möglich zu helfen, aber es möchte auch mit Hilfe der Ergebnisse seine Dienstleistungen objektiv bewerten können. Die Objektivität der Ergebnisse ist hierbei kritisch zu betrachten, da persönliche Einflüsse des Kunden die Bewertung erheblich beeinflussen. Letztendlich

ist es für Unternehmen schwierig einzuschätzen, ob ein sehr negativ bewerteter Tweet tatsächlich objektiv schlimmer ist, als ein weniger negativ bewerteter Tweet. Ebenso kann es sein, dass sich der sehr negativ bewertete Tweet durch persönliche Umstände des Kunden ergibt. Wenn beispielsweise der Kunde ein Produkt zu einem wichtigen Stichtag, zum Beispiel für eine Hochzeit, bestellt und dieses Produkt nicht pünktlich geliefert wird, ist anzunehmen, dass dieser Kunde deutlich verärgerter sein wird als ein Kunde, dessen Lieferung nicht von persönlicher Relevanz war. Trotz unterschiedlich starker Bewertung, die aus dem subjektivem Empfinden resultiert, kann objektiv dieselbe Ursache vorliegen. Einerseits ergibt sich daraus für Unternehmen die Schwierigkeit der Priorisierung der Anfragen. Die Anfragen können einfach chronologisch abgearbeitet werden oder man entscheidet sich dafür, diese anhand des Sentimentwertes zu priorisieren. Andererseits entsteht die Notwendigkeit der Bereinigung von subjektiven Einflüssen, um eine objektive Bewertung der eigenen Dienstleistungen gewährleisten zu können.

Kapitel 6

Fazit

Bereits zu Beginn der Arbeit stand die Frage offen, wie Unternehmen die Informationsflut und Großzahl an Anfragen bearbeiten sollen, um dem Kunden kurzfristig eine zufriedenstellende Rückmeldung zu geben, Durch die Analyse in dieser Masterarbeit wurde aufgezeigt, dass es Methoden und Ansätze gibt mit der Unternehmen die Datenflut verarbeiten und analysieren können. Zusätzlich hat die Ausarbeitung gezeigt, wie wichtig die Meinungen und die Bedürfnisse der Kunden sind. Soziale Medien stellen einen direkten Kommunikationskanal zu Unternehmen dar, wodurch sie viel direkter und vor allem öffentlich kontaktiert werden können. Eine Anfrage eines Kunden ist nicht mehr nur für den Kunden selbst und das Unternehmen sichtbar, sondern geschieht die Kontaktaufnahme öffentlich über soziale Medien kann jeder mitlesen und auch seinen Beitrag dazu schreiben. Die Bewertung der Leistungsqualität eines Unternehmens wird dadurch transparenter. Dabei können sich Probleme schnell zu einem sogenannten *„Shitstorm“* entwickeln. Im Duden wird *Shitstorm* als „Sturm der Entrüstung in einem Kommunikationsmedium des Internets, der zum Teil mit beleidigenden Äußerungen einhergeht“ definiert.[243] Ein solcher Shitstorm ist von Unternehmen zu vermeiden, da das Image des Unternehmens dadurch geschädigt werden kann. Aus Unternehmenssicht kann jedoch nicht auf die Kanäle in den sozialen Medien verzichtet werden, da gerade die junge Generation vermehrt auf diese Kanäle zurückgreift, um Kontakt mit Unternehmen aufzunehmen. Darüber hinaus versuchen Unternehmen alleine schon aus Werbezwecken in sozialen Medien vertreten zu sein. Natürlich hat nicht nur der Kunde einen direkteren Zugang zu Unternehmen, sondern auch umgekehrt ist es der Fall. Unternehmen können ihre Kunden und auch andere potenzielle Kunden besser analysieren, da auch der Nutzer in den sozialen Medien viel von sich preisgibt. Wie das Praxisbeispiel in dieser Ausarbeitung gezeigt hat, werden beispielsweise bei Twitter viele Daten über den Nutzer über die API mit ausgegeben, die dann wiederum

[243] V., 2017.

analysiert werden können.

Zur Analyse dieser Daten sind meist Big-Data-Infrastrukturen notwendig, um die großen und wachsenden Datenmengen verarbeiten zu können. Die Ausarbeitung hat in Kapitel 2 gezeigt, wie die viel genutzte Hadoop-Plattform aufgebaut ist, die dann auch im Praxisbeispiel in Kapitel 5 zum Einsatz kam. Der Markt im Big Data Bereich entwickelt sich sehr schnell und es kommen laufend neue Entwicklungen dazu. Dies trifft sowohl auf die Infrastruktur- und Prozessmanagementkomponenten, als auch auf die Software zur Analyse der Daten zu. Laufend werden Weiterentwicklungen oder Neuentwicklungen auf den Markt gebracht. Diese Entwicklungen lassen sich auch beispielsweise bei Qlik erkennen, die ein Werkzeuge zur Datenvisualisierung und -analyse anbieten, aber immer mehr Fokus auf die Verarbeitung von Daten direkt aus Big-Data-Plattformen legen und auch Echtzeit-Anwendungen ermöglichen.

Um jedoch als Unternehmen Informationen aus den Daten der Nutzer generieren zu können, müssen beispielsweise die Texte der Nutzer analysiert werden können. Kapitel 5 beleuchtet dabei einen möglichen Weg der Analyse - die Sentiment Analyse. Natürlich stellt sich für Unternehmen immer die Frage, wie sie die Daten der Nutzer verwenden können und welche Anwendungsfälle daraus resultieren. Um zu zeigen, dass eine Sentiment Analyse im Unternehmenskontext sinnvoll ist, wurde in Kapitel 5 die Sentiment Analyse dazu verwendet, um den Kundendienst von Amazon im Bezug auf den Twitteraccount *@AmazonHelp* zu verbessern. Das Beispiel hat gezeigt, dass mit einfachen Mitteln eine Auswertung der Twitterdaten anhand der Sentimentklassifizierung vorgenommen und die Arbeit für Unternehmen vereinfacht werden kann. Zusätzlich kann die Sentiment Analyse, wie das Beispiel gezeigt hat, dazu beitragen, dass Ursachen für Probleme von Kunden einfacher kategorisiert und analysiert werden können. Durch die Identifizierung von Objekten auf die sich ein Tweet zum Beispiel bezieht können schnell Trends von Problemen erkannt werden, die möglicherweise auf die gleiche Ursache zurückzuführen sind, wie unter anderem der Ausfall des Logistikdienstleisters oder Probleme mit dem Prime-Service. Durch die frühzeitige Identifizierung von Problem-Trends kann kurzfristig auf Probleme reagiert werden und proaktive Maßnahmen abgeleitet werden. Dies hilft dabei den daraus resultierenden Schaden zu begrenzen.

Zusammenfassend lässt sich sagen, dass eine Sentiment Analyse im Unternehmensumfeld durchaus sinnvoll ist und vor allem dann, wenn das Unternehmen auch in sozialen Netzwerken aktiv ist. Die Sentiment Analyse kann auch für andere Kommunikationswege, wie E-Mails oder Kontaktformulare genutzt werden. Soziale Netzwerke werden für solche Zwecke jedoch immer wichtiger. Im Falle von Amazon könnte zur weiteren Verbesserung das Sentiment-Lexikon noch weiter verfeinert werden. Dies kann dazu genutzt werden, eine höhere Genauigkeit zu erzielen und präzisere Vorhersagen zu treffen. Damit auch andere Sprachen neben Englisch analysiert werden können, kann die Flume-Konfiguration so angepasst werden, dass nur bestimmte oder alle Sprachen von der API bezogen werden. Im Fall des Beispiels müssten zur Analyse und Klassifizierung von anderen Sprachen auch die Hivemall-Funktionen angepasst und entsprechende Lexika erstellt werden. Natürlich können auch wie in Kapitel 4 erläutert Machine-Learning-Ansätze genutzt werden, um die Klassifizierung durchzuführen. Klar ist jedoch, dass ein großer Nutzen und Mehrwert entsteht, wenn die Kommunikation, über soziale Netzwerke, analysiert wird. Es können Arbeitsabläufe in der Bearbeitung der Anfragen vereinfacht werden und die Informationen, die aus der Analyse resultieren, können dazu genutzt werden, um den Kundenservice zu verbessern oder Probleme frühzeitig zu erkennen.

Abschließend ist zu sagen, dass die vorliegende Masterthesis Ansätze und Methoden aufgezeigt hat mit denen Unternehmen in der Lage sind die Daten aus sozialen Netzwerken zu speichern, zu verarbeiten und schließlich zu analysieren. Praktisch wurde dies anhand des Beispiels mit Amazons Kundendienst auf Twitter mit Hilfe einer Sentiment Analyse aufgezeigt. Die aus der Theorie und den praktischen Ergebnissen gewonnenen Erkenntnisse über Herausforderungen, Nutzen, sowie Methoden zur Umsetzung sind universell in vielen Unternehmen einsetzbar und können wie aufgezeigt unter anderem zu einer Verbesserung der Kundenzufriedenheit führen.

Literatur

Aggarwal, Charu C. und ChengXiang Zhai, Hrsg. (2012). *Mining Text Data*. Springer. ISBN: 978-1-4419-8462-3.

Alpaydin, Ethem (2014). *Introduction to Machine Learning*. The MIT Press. 640 S. ISBN: 0262028182.

Ananiadou, Sophia und John McNaught, Hrsg. (2006). *Text mining for biology and biomedicine*. Artech House bioinformatics series. Boston MA, London: Artech House. ISBN: 1-58053-984-X.

Andreas Meier, Michael Kaufmann (2016). *SQL- & NoSQL-Datenbanken*. Springer-Verlag GmbH.

Apache (2013). *HDFS Architecture Guide*. Abgerufen am 11.01.2017. Apache Software Foundation. URL: `https://hadoop.apache.org/docs/r1.2.1/hdfs_user_guide.pdf`.

— (2016). *What Is Apache Hadoop?* Abgerufen am 10.01.2017. Apache Software Foundation. URL: `http://hadoop.apache.org/`.

API, Twitter Search (2017). *Twitter Search API*. Abgerufen am: 13.04.2017. URL: `https://dev.twitter.com/rest/public/search`.

Baccianella, Stefano, Andrea Esuli und Fabrizio Sebastiani (2010). „SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining.“ In: *LREC*. Bd. 10, S. 2200–2204.

Barczok, Achim (2008). „Die Welt in 140 Zeichen. Twitter bietet ganz neue Möglichkeiten, sich mit Freunden zu vernetzen“. In: *c't* 14, S. 86–87.

Bättig, Daniel (2014). *Angewandte Datenanalyse: Der Bayes'sche Weg*. Springer Spektrum. ISBN: 978-3-662-43393-5.

Bodendorf, Freimut (2006). *Daten- und Wissensmanagement*. Springer-Verlag Berlin Heidelberg.

Borod, J.C. (2000). *The Neuropsychology of Emotion*. Series in Affective Science. Oxford University Press. ISBN: 9780198027409.

Bößwetter, Daniel (2009). „Spaltenorientierte Datenbanken“. In: *Informatik-Spektrum* 33.1, S. 61–65. DOI: `10.1007/s00287-009-0401-2`.

Boyd, Ryan (2016). *Getting Started with OAuth 2.0*. O'Reilly UK Ltd. 66 S. ISBN: 1449311601.

Bramer, Max (2013). *Principles of Data Mining*. Springer. 456 S. ISBN: 1447148835.

Bulygo, Zach (2013). *How Netflix Uses Analytics To Select Movies, Create Content, and Make Multimillion Dollar Decisions*. Abgerufen am 24.11.2016: URL: https://blog.kissmetrics.com/how-netflix-uses-analytics/.

Buyya, R., R.N. Calheiros und A.V. Dastjerdi (2016). *Big Data: Principles and Paradigms*. Elsevier Science. ISBN: 9780128093467.

Celko, Joe (2013). *Joe Celko's Complete Guide to NoSQL: What Every SQL Professional Needs to Know about Non-Relational Databases*. Morgan Kaufmann. ISBN: 0124072208.

Chang, Fay u. a. (2008). „Bigtable: A Distributed Storage System for Structured Data". In: *ACM Trans. Comput. Syst.* 26.2, 4:1–4:26. ISSN: 0734-2071. DOI: 10.1145/1365815.1365816. URL: http://doi.acm.org/10.1145/1365815.1365816.

Charniak, Eugene u. a. (1993). „Equations for part-of-speech tagging". In: *Proceedings of the eleventh national conference on Artificial intelligence*. AAAI Press, S. 784–789.

Cutting, Doug u. a. (1992). „A practical part-of-speech tagger". In: *Proceedings of the third conference on Applied natural language processing*. Association for Computational Linguistics, S. 133–140.

Dean, Jeffrey und Sanjay Ghemawat (2004). „MapReduce". In: *Communications of the ACM* 51.1, S. 107. DOI: 10.1145/1327452.1327492.

Decker, K.M. und S. Focardi (1995). *Technology Overview: A Report on Data Mining*. CSCS-ETH.

deRoos, Dirk u. a. (2014). *Hadoop For Dummies*. John Wiley & Sons Inc. 408 S. ISBN: 1118607554.

Economist, The und SAP (2010). *Data, data everywhere*. English. Abgerufen am 01.12.2016. URL: https://ai.arizona.edu/sites/ai/files/MIS510/data_data_everywhere_sap_and_the_economist.pdf.

Edlich, Prof. Dr. Stefan (2016). *List of NoSQL Databases*. Abgerufen am 21.12.2016. URL: http://nosql-database.org/.

Edlich, Prof. Dr. Stefan u. a. (2010). *NoSQL: Einstieg in die Welt nichrelationaler Web 2.0 Datenbanken*. Hanser Verlag.

EMC (2014). *Digitales Universum explodiert durch Sensordaten*. Abgerufen am 31.08.2016. URL: http://germany.emc.com/about/news/press/2014/20140409-01.htm.

Fellbaum, Christiane (1998). *Wordnet: An Electronic Lexical Database*. MIT PR. 449 S. ISBN: 026206197X.

Foundation, Apache Software (2012). *Flume User Guide*. Abgerufen am 25.01.2017. Apache Software Foundation. URL: http://flume.apache.org/FlumeUserGuide.html.

Fowler, Adam (2015). *NoSQL For Dummies*. John Wiley & Sons Inc. 456 S. ISBN: 1118905741.

Garg, Vikram und Sharan Kumar (2015). *Mastering Social Media Mining with R*. Packt Publishing. 248 S. ISBN: 1784396311.

Garreta, Raul und Guillermo Moncecchi (2013). *Learning scikit-learn: Machine Learning in Python*. Packt Publishing. 118 S. ISBN: 1783281936.

Ghemawat, Sanjay, Howard Gobioff und Shun-Tak Leung (2003). „The Google file system". In: *Proceedings of the nineteenth ACM symposium on Operating systems principles - SOSP '03*. Association for Computing Machinery (ACM). DOI: 10.1145/945445.945450.

Greene, B.B. und G.M. Rubin (1971). *Automatic Grammatical Tagging of English*. Department of Linguistics, Brown University.

Haluschak, Bernhard (2016). *Die wichtigsten IT-Herausforderungen für das Datenmanagement*. Abgerufen am 29.05.2017. Computerwoche. URL: https://www.computerwoche.de/a/die-wichtigsten-it-herausforderu
fuer-das-datenmanagement,3328567.

Hand, David J (2001). *Principles of Data Mining*. MIT University Press Group Ltd. ISBN: 026208290X.

Harrison, Guy (2015). *Next Generation Databases: NoSQL, NewSQL, and Big Data*. Apress.

Holmes, Alex (2012). *Hadoop in Practice*. MANNING PUBN. 536 S. ISBN: 1617290238.

Hurwitz, Judith (2013). *Big Data For Dummies*. Hrsg. von Alan Nugent; Marcia Kaufman; Fern Halper; Dan Kirsch. John Wiley & Sons Inc. ISBN: 1118504224.

IBM (2013). *The Four V's of Big Data*. Aufgerufen am 01.09.2016. URL: http://www.ibmbigdatahub.com/infographic/four-vs-big-data.

Jordan, Franz (2014). *150 Mio Produkte auf Amazon DE 280 Mio in USA*. Abgerufen am 08.04.2017. URL: https://marketplace-analytics.de/blog-amazon-sortimentgroesse-laender-vergleich.

Kao, Anne (2006). *Natural Language Processing and Text Mining*. Hrsg. von Stephen R. Poteet. Springer-Verlag GmbH. ISBN: 184628175X.

Kennedy, Alistair und Diana Inkpen (2006). „Sentiment classification of movie reviews using contextual valence shifters". In: *Computational intelligence* 22.2, S. 110–125.

King, Gary (o.D.). *Preface: Big Data Is Not About The Data!* Abgerufen am 02.12.2017. Institute for Quantitative Social Science - Harvard University. URL: http://gking.harvard.edu/files/gking/files/prefaceorbigdataisnotaboutthedata_1.pdf.

Klein, Franziska (o. J.). *Die Datenflut und was sie für Unternehmen bedeuten*. Abgerufen am 29.05.2017. Connected Solutions. URL: https://connected-solutions.de/die-datenflut-und-was-sie-fuer-unternehmen-bedeuten/.

Lam, Chuck (2011). *Hadoop in Action*. Manning. ISBN: 1935182196.

Liu, Bing (2007). *Web Data Mining - Exploring Hyperlinks, Contents, and Usage Data*. Springer Verlag.

Liu, Huan, Mohammad-Ali Abbasi und Reza Zafarani (2014). *Social Media Mining*. Cambridge University Press. 332 S. ISBN: 1107018854.

Lukoianova, Tatiana und Victoria L Rubin (2014). „Veracity roadmap: Is big data objective, truthful and credible?“ In: *Advances in Classification Research Online*.

Manney, Darin (2017). *Amazon Com Announces Fourth Quarter Sales*. Abgerufen am 08.04.2017. URL: http://phx.corporate-ir.net/phoenix.zhtml?c=97664&p=irol-newsArticle&ID=2241835.

Manning, Christopher S. (1999). *Foundations of Statistical Natural Language Processing*. MIT University Press Group Ltd. ISBN: 0262133601.

Markus Hofmann, Andrew Chisholm (2016). *Text Mining and Visualization*. CRC Press.

McCreary, Dan und Ann Kelly (2013). *Making Sense of NoSQL*. Manning. ISBN: 1617291072.

Meier, Andreas (2016). „Zur Nutzung von SQL- und NoSQL-Technologien“. In: *HMD Praxis der Wirtschaftsinformatik* 53.4, S. 415–427. DOI: 10.1365/s40702-016-0225-x.

Mohanty, Soumendra, Madhu Jagadeesh und Harsha Srivatsa (2013). *Big Data Imperatives - Enterprise Big Data Warehouse, BI Implementations and Analytics*. Apress.

Mueller, John Paul und Luca Massaron (2016). *Machine Learning For Dummies*. John Wiley & Sons Inc. 432 S. ISBN: 1119245516.

Murthy, Arun C. u. a. (2014). *Apache Hadoop YARN*. Addison Wesley. 400 S. ISBN: 0321934504.

Nielsen, Finn Årup (2011). *AFINN*. Abgerufen am 25.04.2017. URL: http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010.

Ohlhorst, Frank J. (2012). *Big Data Analytics: Turning Big Data Into Big Money*. JOHN WILEY & SONS INC. ISBN: 1118147596.

Pang, Bo und Lillian Lee (2008). „Opinion Mining and Sentiment Analysis“. In: *Found. Trends Inf. Retr.* 2.1-2, S. 1–135. ISSN: 1554-0669. DOI: 10.1561/1500000011. URL: http://dx.doi.org/10.1561/1500000011.

Pang, Bo, Lillian Lee und Shivakumar Vaithyanathan (2002). „Thumbs Up?: Sentiment Classification Using Machine Learning Techniques“. In: *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*. EMNLP '02. Stroudsburg, PA, USA: Association for Computational Linguistics, S. 79–86. DOI: 10.3115/1118693.1118704. URL: https://doi.org/10.3115/1118693.1118704.

Perrin, Matthieu, Achour Mostéfaoui und Claude Jard (2016). „Causal Consistency: Beyond Memory“. In: *CoRR* abs/1603.04199.

Plattner, Prof. Dr. Hasso (2013). *Big Data*. Abgerufen am 01.09.2016. URL: http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/daten-wissen/Datenmanagement/Datenmanagement--Konzepte-des/Big-Data.

Polanyi, Livia und Annie Zaenen (2006). „Contextual valence shifters“. In: *Computing attitude and affect in text: Theory and applications*. Springer, S. 1–10.

Popescu, Ana-Maria und Orena Etzioni (2007). „Extracting product features and opinions from reviews“. In: *Natural language processing and text mining*. Springer, S. 9–28.

Porter, M. F. (1997). „An Algorithm for Suffix Stripping“. In: *Readings in Information Retrieval*. Hrsg. von Karen Sparck Jones und Peter Willett. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. Kap. An Algorithm for Suffix Stripping, S. 313–316. ISBN: 1-55860-454-5.

Pozzi, Federico u. a. (2016). *Sentiment Analysis in Social Networks*. Elsevier LTD, Oxford. ISBN: 0128044128.

Prajapati, Vignesh (2013). *Big Data Analytics with R and Hadoop*. Packt Publishing.

Rao, Srikumar S. (1998). *Diaper beer syndrome*. Abgerufen am 12.09.2016. Forbes. URL: http://www.forbes.com/forbes/1998/0406/6107128a.html.

Rapp, Prof. Dr. Reinhold (2016). *Retail Revolution: Die 5 Erfolgsfaktoren*. Abgerufen am 10.05.2017. URL: https://www.zukunftsinstitut.de/artikel/08-retail-revolution/01-longreads/retail-revolution-die-5-erfolgsfaktoren/.

Richard Heimann, Nathan Danneman (2014). *Social Media Mining with R*. Packt Publishing. 122 Seiten. ISBN: 1783281774.

Russell, Matthew (2013). *Mining the Social Web*. O'Reilly Media, Inc, USA. 448 S. ISBN: 1449367615.

Sammut, Claude und Geoffrey I. Webb, Hrsg. (2011). *Encyclopedia of Machine Learning*. Springer Verlag. 1031 S. ISBN: 0387307680.

Schieber, Andreas, Andreas Hilbert und Carsten Stillich (2012). „Identifikation und Analyse von ironischen und sarkastischen Kundenrezensionen im Web". In: *Multikonferenz Wirtschaftsinformatik 2012 : Tagungsband der MKWI 2012*.

Schmidt, Jan-Hinrik (2013). *Social Media*. Gabler, Betriebswirt.-Vlg. 108 S. ISBN: 3658020954.

Simon, Phil (2014). *Big Data Lessons from Netflix*. Abgerufen am 24.11.2016: Wired. URL: https://www.wired.com/insights/2014/03/big-data-lessons-netflix/.

Sitto, Kevin und Marshall Presser (2015). *Field Guide to Hadoop*. O'Reilly UK Ltd. 130 S. ISBN: 1491947934.

Terry, D. B. u. a. (1994). „Session guarantees for weakly consistent replicated data". In: *Proc. 3rd Int. Conf. Parallel and Distributed Information Systems*, S. 140–149. DOI: 10.1109/PDIS.1994.331722.

Tsur, Oren, Dmitry Davidov und Ari Rappoport (2010). „ICWSM — A Great Catchy Name: Semi-Supervised Recognition of Sarcastic Sentences in Online Product Reviews". In: *Proceedings of the Fourth International Conference on Weblogs and Social Media (ICWSM-2010)*. Hrsg. von Marti Hearst, William Cohen und Samuel Gosling. The AAAI Press, Menlo Park, California. URL: http://www.aaai.org/ocs/index.php/ICWSM/ICWSM10/paper/view/1495/1851.

Turkington, Garry (2013). *Hadoop Beginner's Guide*. Packt Publishing.

Turney, Peter D. (2002). „Thumbs Up or Thumbs Down?: Semantic Orientation Applied to Unsupervised Classification of Reviews". In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. ACL '02. Philadelphia, Pennsylvania: Association for Computational Linguistics, S. 417–424. DOI: 10.3115/1073083.1073153. URL: http://dx.doi.org/10.3115/1073083.1073153.

Twitter (2017a). *Twitter - API Overview*. Abgerufen am 13.04.2017. URL: https://dev.twitter.com/overview/api.

— (2017b). *Twitter Streaming API*. Abgerufen am: 13.04.2017. URL: https://dev.twitter.com/streaming/overview.

Upbin, Bruce (2012). *The Web Is Much Bigger And Smaller Than You Think*. Abgerufen am 01.12.2017. Forbes. URL: http://www.forbes.com/sites/ciocentral/2012/04/24/the-web-is-much-bigger-and-smaller-than-you-think/#42980da645de.

V., o. (2015). *BASE*. Abgerufen am 21.12.2016. FH Köln Campus Gummersbach. URL: http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken/BASE.

— (2017). *Shitstorm*. Abgerufen am: 26.05.2017. Duden. URL: http://www.duden.de/rechtschreibung/Shitstorm.

Vogels, Werner (2007). *Eventually Consistent*. Abgerufen am 03.01.2017. URL: http://www.allthingsdistributed.com/2007/12/eventually_consistent.html.

Warden, Pete (2011). *Big Data Glossary*. O'REILLY & ASSOC INC. 62 S. ISBN: 1449314597.

Weiss, Sholom M. (2004). *Text Mining*. Hrsg. von Frederick Damerau Nitin Indurkhya T. Zhang. Springer-Verlag GmbH. ISBN: 0387954333.

Weiss, Sholom M., Nitin Indurkhya und Tong Zhang (2015). *Fundamentals of Predictive Text Mining*. Springer-Verlag GmbH. ISBN: 144716749X.

White, Tom (2016). *Hadoop The Definitive Guide*. O Reilly UK Ltd. ISBN: 1491901632.

Wiese, Lena (2015). *Advanced Data Management: For SQL, NoSQL, Cloud and Distributed Databases*. De Gruyter Textbook. De Gruyter. ISBN: 9783110441413.

Wilcock, Graham (2009). *Introduction to Linguistic Annotation and Text Analytics*. Morgan & Claypool Publishers. 160 S. ISBN: 1598297384.

Wilson, Theresa, Janyce Wiebe und Paul Hoffmann (2005). „Recognizing Contextual Polarity in Phrase-level Sentiment Analysis". In: *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. HLT '05. Vancouver, British Columbia, Canada: Association for Computational Linguistics, S. 347–354. DOI: 10.3115/1220575.1220619. URL: http://dx.doi.org/10.3115/1220575.1220619.

Wu, Steven u. a. (2016). *Evolution of the Netflix Data Pipeline*. Abgerufen am 24.11.2016: URL: http://techblog.netflix.com/2016/02/evolution-of-netflix-data-pipeline.html.

Zafarani, Reza, Mohammad Ali Abbasi und Huan Liu (2014). *Social Media Mining: An Introduction*. Cambridge University Press. ISBN: 978-1-107-01885-3.

Zarrella, Dan (2010). *Social Media Marketing*. O'Reilly Verlag. ISBN: 978-3-89721-657-0.

Ziegler, Cai (2006a). „Die Vermessung der Meinung". In: *iX* 10, S. 106–109.

Ziegler, Cai (2006b). „Stummer Wächter“. In: *iX* 4, S. 116–119.

Anhang A

Flume Konfiguration

```
agent.sources = PublicStream
agent.channels = MemCh
agent.sinks = HDFS

agent.sources.PublicStream.type =
mmartsen.flume.sources.twitter.TwitterSource
agent.sources.PublicStream.channels = MemCh
agent.sources.PublicStream.consumerKey = **Key**
agent.sources.PublicStream.consumerSecret = **Secret**
agent.sources.PublicStream.accessToken = **Token**
agent.sources.PublicStream.accessTokenSecret = **Secret**
agent.sources.PublicStream.keywords = @AmazonHelp
agent.sources.PublicStream.language = en


agent.sinks.HDFS.channel = MemCh
agent.sinks.HDFS.type = hdfs
agent.sinks.HDFS.hdfs.path =
hdfs://localhost:8020/user/Hadoop/twitter_data/
agent.sinks.HDFS.hdfs.fileType = DataStream
agent.sinks.HDFS.hdfs.filePrefix = TwitterPublicStream
agent.sinks.HDFS.hdfs.writeFormat = Text
agent.sinks.HDFS.hdfs.inUsePrefix = _
agent.sinks.HDFS.hdfs.maxOpenFiles = 10
agent.sinks.HDFS.hdfs.batchSize = 1000
agent.sinks.HDFS.hdfs.rollSize = 0
agent.sinks.HDFS.hdfs.rollCount = 10000

agent.channels.MemCh.type = memory
agent.channels.MemCh.capacity = 100000
```

```
agent.channels.MemCh.transactionCapacity = 1000
```

Anhang B

Hive Code

Hinzufügen der Jar-Dateien

```
add jar /usr/lib/hive/lib/hive-serdes-1.0-SNAPSHOT.jar;
add jar
/tmp/hivemall-core-0.4.2-rc.2-with-dependencies.jar;
```

Anlegen der Tweets-Tabelle

```
DROP table tweets;

CREATE EXTERNAL TABLE tweets (
        id BIGINT,
        created_at STRING,
        source STRING,
        favorited BOOLEAN,
        retweeted_status STRUCT<
                text:STRING,
                user:STRUCT<
                        screen_name:STRING,
                        name:STRING>,
        retweet_count:INT>,
        entities STRUCT<
        urls:ARRAY<STRUCT<
        expanded_url:STRING>>,
        user_mentions:ARRAY<STRUCT<
                screen_name:STRING,
                name:STRING>>,
        hashtags:ARRAY<STRUCT<text:STRING>>>,
        text STRING,
        user STRUCT<
                location:STRING,
                screen_name:STRING,
```

```
                name:STRING,
                friends_count:INT,
                followers_count:INT,
                statuses_count:INT,
                verified:BOOLEAN,
                utc_offset:INT,
                time_zone:STRING>,
        in_reply_to_screen_name STRING
)
ROW FORMAT SERDE 'com.cloudera.hive.serde.JSONSerDe'
LOCATION '/user/Hadoop/twitter_data';
```

Anlegen des Views tweets_user

```
create or replace view tweets_user as
select
        id,
        user.location as location,
        user.followers_count as followers_count,
        user.friends_count as friend_count,
        user.name as username,
        user.screen_name as screen_name,
        user.statuses_count as statuses_count,
        user.time_zone as time_zone,
        user.utc_offset as utc_offset,
        user.verified as verified
from tweets;
```

Anlegen des Views tweets_retweeted

```
create or replace view tweets_retweeted
as select
        id,
        retweeted_status.retweet_count as retweet_count,
        retweeted_status.text as retweet_text,
        retweeted_status.user.name as user_name
from tweets;
```

Anlegen des Views tweets_entities

```
create or replace view tweets_entities
as select
```

```
        id,
        entities.hashtags as hashtags,
        entities.user_mentions as user_mentions,
        entities.urls as urls
from tweets;
```

Anlegen der Dictionary-Tabelle und Laden des Wörterbuchs

```
drop table dictionary;
create table dictionary(
        word string,
        rating int
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

LOAD DATA INPATH '/AFINN.txt' into TABLE dictionary;
```

Anlegen der Hivemall-Funktionen

```
-------------------------------------------------------
-- Hivemall: Hive scalable Machine Learning Library
-------------------------------------------------------

drop temporary function hivemall_version;
create temporary function hivemall_version
        as 'hivemall.HivemallVersionUDF';

--------------------
-- misc functions --
--------------------

drop temporary function max_label;
create temporary function max_label
        as 'hivemall.ensemble.MaxValueLabelUDAF';

drop temporary function maxrow;
create temporary function maxrow
        as 'hivemall.ensemble.MaxRowUDAF';

drop temporary function argmin_kld;
create temporary function argmin_kld
```

```
        as 'hivemall.ensemble.ArgminKLDistanceUDAF';

-----------------------
-- scaling functions --
-----------------------

drop temporary function rescale;
create temporary function rescale
        as 'hivemall.ftvec.scaling.RescaleUDF';

drop temporary function zscore;
create temporary function zscore
        as 'hivemall.ftvec.scaling.ZScoreUDF';

drop temporary function l2_normalize;
create temporary function l2_normalize
        as 'hivemall.ftvec.scaling.L2NormalizationUDF';

-----------------------------------
-- Feature engineering functions --
-----------------------------------

drop temporary function amplify;
create temporary function amplify
        as 'hivemall.ftvec.amplify.AmplifierUDTF';

drop temporary function rand_amplify;
create temporary function rand_amplify
        as 'hivemall.ftvec.amplify.RandomAmplifierUDTF';

drop temporary function add_bias;
create temporary function add_bias
        as 'hivemall.ftvec.AddBiasUDF';

drop temporary function sort_by_feature;
create temporary function sort_by_feature
        as 'hivemall.ftvec.SortByFeatureUDF';

drop temporary function extract_feature;
```

```
create temporary function extract_feature
        as 'hivemall.ftvec.ExtractFeatureUDF';

drop temporary function extract_weight;
create temporary function extract_weight
        as 'hivemall.ftvec.ExtractWeightUDF';

drop temporary function add_feature_index;
create temporary function add_feature_index
        as 'hivemall.ftvec.AddFeatureIndexUDF';

drop temporary function feature;
create temporary function feature
        as 'hivemall.ftvec.FeatureUDF';

drop temporary function feature_index;
create temporary function feature_index
        as 'hivemall.ftvec.FeatureIndexUDF';


--------------------------
-- ftvec/text functions --
--------------------------

drop temporary function tf;
create temporary function tf
as 'hivemall.ftvec.text.TermFrequencyUDAF';



------------------------------
-- Text processing functions --
------------------------------

drop temporary function tokenize;
create temporary function tokenize
         as 'hivemall.tools.text.TokenizeUDF';

drop temporary function is_stopword;
```

```
create temporary function is_stopword
        as 'hivemall.tools.text.StopwordUDF';

drop temporary function split_words;
create temporary function split_words
        as 'hivemall.tools.text.SplitWordsUDF';

drop temporary function normalize_unicode;
create temporary function normalize_unicode
        as 'hivemall.tools.text.NormalizeUnicodeUDF';

drop temporary function base91;
create temporary function base91
        as 'hivemall.tools.text.Base91UDF';

drop temporary function unbase91;
create temporary function unbase91
        as 'hivemall.tools.text.Unbase91UDF';


--------------------
-- General Macros --
--------------------

create temporary macro java_min(x DOUBLE, y DOUBLE)
reflect("java.lang.Math", "min", x, y);

create temporary macro max2(x DOUBLE, y DOUBLE)
if(x>y,x,y);

create temporary macro min2(x DOUBLE, y DOUBLE)
if(x<y,x,y);

create temporary macro rand_gid(k INT)
floor(rand()*k);

create temporary macro rand_gid2(k INT, seed INT)
floor(rand(seed)*k);
```

```
--------------------------
-- Statistics functions --
--------------------------

create temporary macro idf(df_t DOUBLE, n_docs DOUBLE)
log(10, n_docs / max2(1,df_t)) + 1.0;

create temporary macro
tfidf(tf FLOAT, df_t DOUBLE, n_docs DOUBLE)
tf * (log(10, n_docs / max2(1,df_t)) + 1.0);
```

Anlegen des Views tokenized

```
create or replace view tokenized
as select
        id,
        tokenize(text, true) as text
from tweets;
```

Anlegen des Views tokenized_exploded

```
create or replace view tokenized_exploded
as select
        id,
        word
from tokenized LATERAL VIEW explode(text) w as word
where not is_stopword(word);
```

Anlegen des Views word_join

```
create or replace view word_join
as select
        tokenized_exploded.id,
        tokenized_exploded.word,
        dictionary.rating
from tokenized_exploded
LEFT OUTER JOIN dictionary
ON(tokenized_exploded.word =dictionary.word);
```

Anlegen des Views id_rating

```
create or replace view id_rating as
select
        word_join.id,
        avg(word_join.rating) as rating
from word_join
group by word_join.id
order by rating DESC;
```

Anlegen des Views text_rating

```
create or replace view text_rating
as select
        tweets.id,
        tweets.text,
        id_rating.rating
from tweets
left join id_rating on(tweets.id=id_rating.id);
```

Zufällige 100 Tweets-Sample

```
select
        *
from text_rating
distribute by rand()
sort by rand()
limit 100;
```

Abgeänderte Stopwords-Funktion

```
package hivemall.tools.text;

import static hivemall.utils.hadoop.WritableUtils.val;

import java.util.Arrays;

import org.apache.hadoop.hive.ql.exec.Description;
import org.apache.hadoop.hive.ql.exec.UDF;
import org.apache.hadoop.hive.ql.udf.UDFType;
import org.apache.hadoop.io.BooleanWritable;

@Description(name = "is_stopword",
```

```
value = "_FUNC_(string word) -
Returns whether English stopword or not")
@UDFType(deterministic = true, stateful = false)
public final class StopwordUDF extends UDF {

private static final String[] stopwords;
static {
        stopwords = new String[] {"i", "me",
        "my", "myself", "we",
        "our", "ours", "ourselves",
        "you", "your", "yours", "yourself",
        "yourselves", "he", "him", "his", "himself",
        "she", "her", "hers", "herself", "it", "its",
        "itself", "they", "them", "their",
        "theirs", "themselves", "what", "which",
        "who", "whom", "this", "that", "these",
        "those", "am", "is", "are", "was", "were",
        "be", "been", "being", "have", "has",
        "had", "having", "do", "does",
        "did", "doing", "a",
        "an", "the", "and", "but",
        "if", "or", "because", "as",
        "until", "while", "of",
        "at", "by", "for", "with",
        "about", "against", "between", "into",
        "through", "during", "before", "after",
        "above", "below", "to", "from",
        "up", "down", "in",
        "out", "on", "off", "over",
        "under", "again", "further", "then", "once",
        "here", "there", "when", "where",
        "why", "how", "all", "any",
        "both", "each", "few",
        "more", "most", "other", "some",
        "such", "no", "nor", "amazon", "@amazon",
        "@amazonhelp", "@amazonin", "https", "//t",
        "rt", "#", "@amazonuk", "co/qupk3ctiql",
        "@slpng_giants", "@jeffbezos", "@gcbrooks",
        "only", "own", "same", "so",
```

```
        "than", "too", "very",
        "s", "t", "can", "will", "just",
        "don", "should", "now"};
        Arrays.sort(stopwords);
}

public BooleanWritable evaluate(String word) {
        return val(Arrays.binarySearch(
        stopwords, word) >= 0);
        }

}
```

Anhang C

Qlik Sense

Anlegen der Verbindung zu Hive

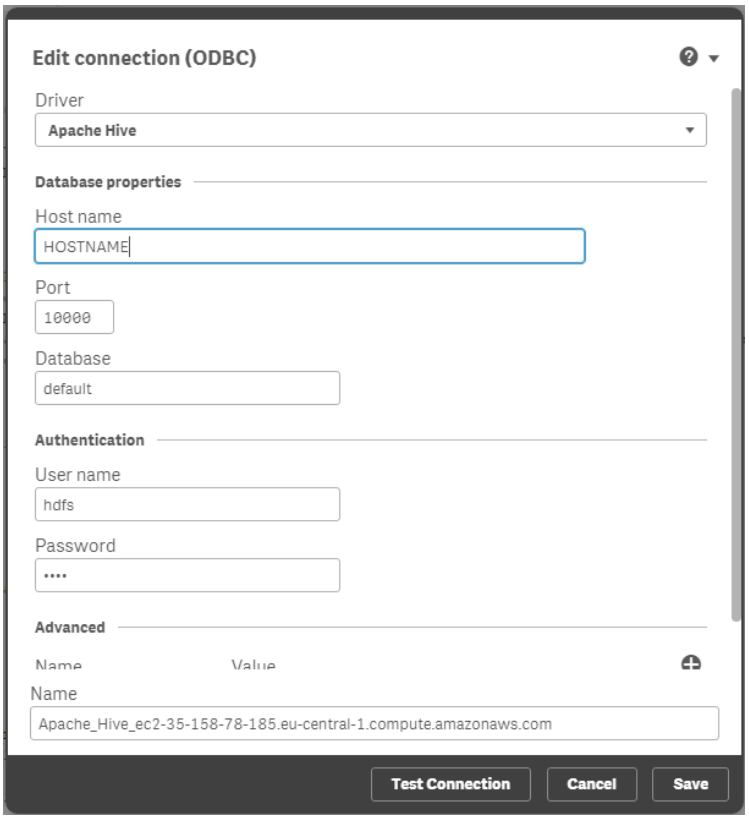

ABBILDUNG C.1: Qlik Sense Hive-Verbindung

Qlik Sense Ladeskript

```
SET ThousandSep='.';
SET DecimalSep=',';
SET MoneyThousandSep='.';
SET MoneyDecimalSep=',';
SET MoneyFormat='#.##0,00 $;-#.##0,00 $';
SET TimeFormat='hh:mm:ss';
SET DateFormat='DD.MM.YYYY';
SET TimestampFormat='DD.MM.YYYY hh:mm:ss[.fff]';
```

```
SET FirstWeekDay=0;
SET BrokenWeeks=0;
SET ReferenceDay=4;
SET FirstMonthOfYear=1;
SET CollationLocale='de-DE';
SET CreateSearchIndexOnReload=1;
SET MonthNames='Jan;Feb;Mrz;Apr;Mai;
Jun;Jul;Aug;Sep;Okt;Nov;Dez';
SET LongMonthNames='Januar;Februar;März;April;
        Mai;Juni;Juli;August;September;
        Oktober;November;Dezember';
SET DayNames='Mo;Di;Mi;Do;Fr;Sa;So';
SET LongDayNames='Montag;Dienstag;
        Mittwoch;Donnerstag;
        Freitag;Samstag;Sonntag';


LIB CONNECT TO 'Apache_Hive.amazonaws.com';

tbl_tweet_stamm:
LOAD
        id,
        created_at,
        favorited,
        text,
        in_reply_to_screen_name;

[tweets_qlik]:
SELECT
        id,
        [created_at],
        favorited,
        text,
        [in_reply_to_screen_name]
FROM [default].[tweets_qlik];

tbl_tweets_stamm:
Load
        *,
```

```
        if(SubStringCount(text,'prime')>0,1,0)
         as Prime_Flag,
        if(SubStringCount(text,'account')>0,1,0)
         as Account_Flag,
        if(SubStringCount(text,'delivery')>0,1,0)
         as Delivery_Flag,
        if(SubStringCount(text,'package')>0,1,0)
         as Package_Flag,
        if(SubStringCount(text,'order')>0,1,0)
         as Order_Flag,
        if(SubStringCount(text,'money')>0,1,0)
         as Money_Flag,
        if(SubStringCount(text,'shipping')>0,1,0)
         as Shipping_Flag,
        if(SubStringCount(text,'service')>0,1,0)
         as Service_Flag

Resident tweets_qlik;

Drop table [tweets_qlik];


LIB CONNECT TO 'Apache_Hive.amazonaws.com';

tbl_rating_tmp:
LOAD
        id,
        if(IsNull(rating),0,rating) as rating;

[qlik_id_rating]:
SELECT
        id,
        rating
FROM [default].[qlik_id_rating];

tbl_rating:
Load
        *,
        if(rating>0,'pos',if(rating<0,'neg','neutral'))
```

```
         as Sentiment
Resident qlik_id_rating
where id <> 'Null' or id <> ' ';

drop table qlik_id_rating;

LIB CONNECT TO 'Apache_Hive.amazonaws.com';

LOAD
        id,
        word;

[qlik_words]:
SELECT
        id,
        word
FROM [default].[qlik_words];

LIB CONNECT TO 'Apache_Hive.amazonaws.com';

LOAD
        id,
        location,
        followers_count,
        friend_count,
        username,
        screen_name,
        statuses_count,
        time_zone,
        utc_offset,
        verified;

[qlik_tweets_user]:
SELECT
        id,
        [location],
        [followers_count],
        [friend_count],
        username,
```

```
        [screen_name],
        [statuses_count],
        [time_zone],
        [utc_offset],
        verified
FROM [default].[qlik_tweets_user];
```

Qlik Sense Datenmodell

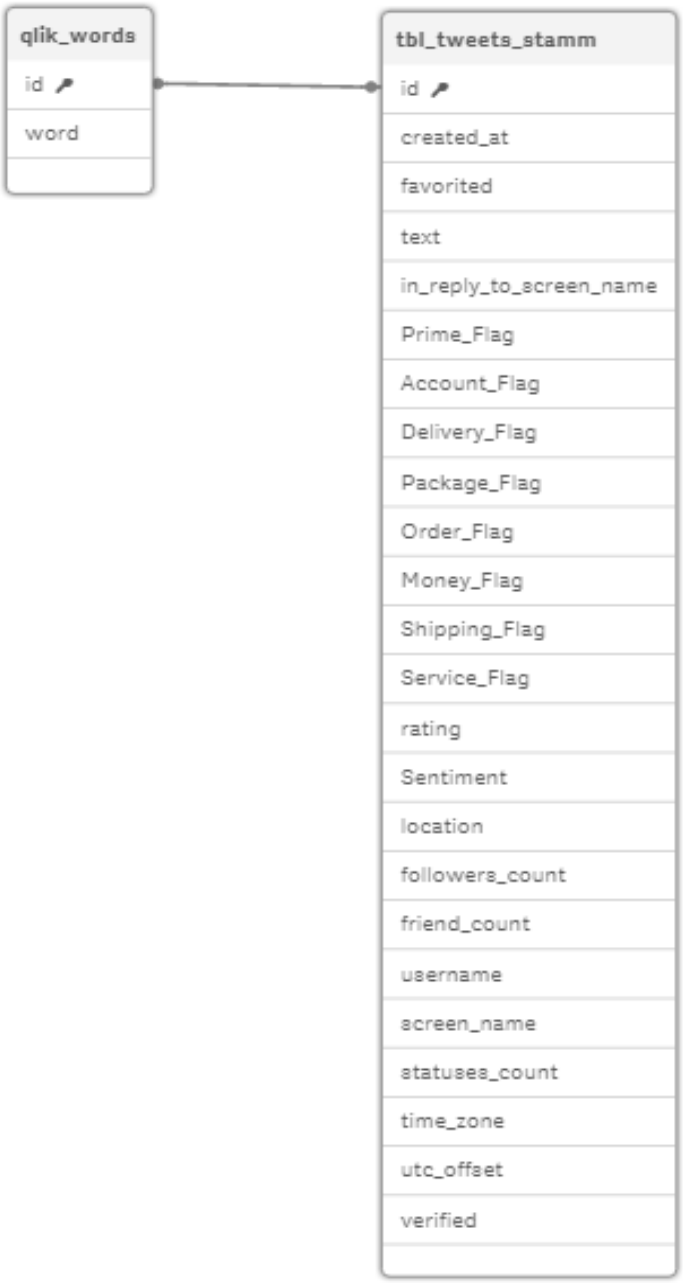

ABBILDUNG C.2: Datenmodell in Qlik Sense

Dashboard - Amazon Twitter Analyse

ABBILDUNG C.3: Dashboard - Amazon Twitter Analyse

Wordcloud vor der Optimierung

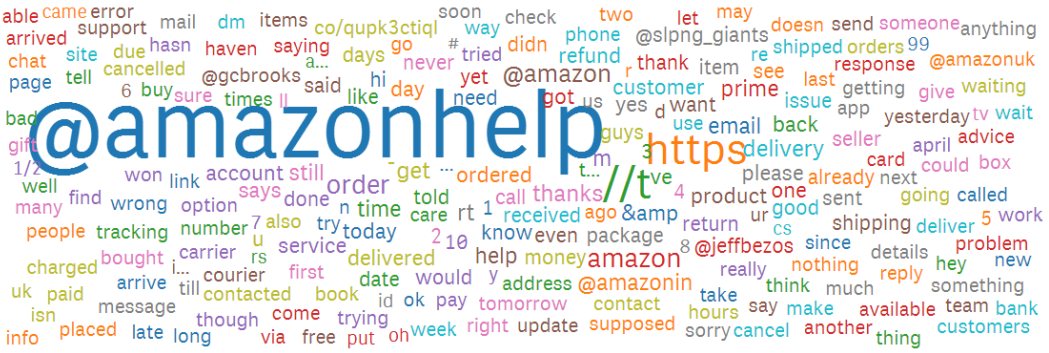

ABBILDUNG C.4: Wordcloud vor der Optimierung

Wordcloud nach der Optimierung

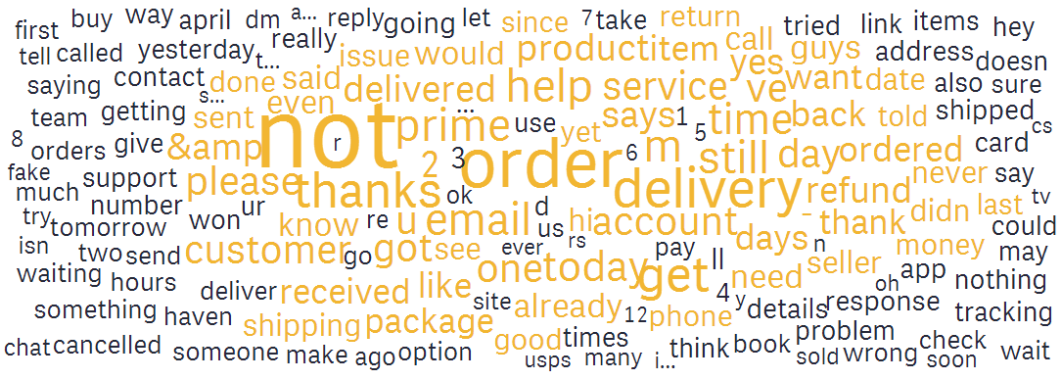

ABBILDUNG C.5: Wordcloud nach der Optimierung

Anhang D

Optimierung der Analyse

TABELLE D.1: Änderungen am Sentimentlexikon

Wort	**in AFINN vorhanden?**	**Wert Vorher**	**Wert Nachher**	**Begründung**
still	nein	-	-1	Meist genutzt weil die Leute auf ihre Lieferung warten, wie z.B. "package is still not here"
rude	nein	-	-2	Vorher nicht vorhanden und wird genutzt wenn Mitarbeiter unfreundlich waren.
rudest	nein	-	-2	Vorher nicht vorhanden und wird genutzt wenn Mitarbeiter unfreundlich waren.
complaint	nein	-	-2	Das Wort taucht hauptsächlich in negativen Tweets auf.
waiting	nein	-	-1	Ähnlich wie 2, da meist auf eine Lieferung oder Antwort gewartet wird.
free	ja	1	-	Entfernt, da es einen positiven Wert hatte, es wird jedoch meist bei "free shipping"genutzt, also rein objektiv.
joke	ja	2	-2	Nur in negativen Tweets.
wish	ja	1	-	Bezieht sich meist auf die Wish List und ist entsprechend objektiv
gift	ja	2	-	Bezieht sich meist auf die Amazon Gift Card und ist entsprechend objektiv.
crappy	nein	-	-3	Vorher nicht vorhanden.
late	nein	-	-1	Meist weil eine Lieferung zu spät kommt.
later	nein	-	-1	Meist weil eine Lieferung zu spät kommt.

TABELLE D.1: Änderungen am Sentimentlexikon - Fortsetzung

Wort	**in AFINN vorhanden?**	**Wert Vorher**	**Wert Nachher**	**Begründung**
latest	nein	-	-1	Meist weil eine Lieferung zu spät kommt.
escalated	nein	-	-2	Kunde hat den Fall eskaliert.
escalate	nein	-	-1	Kunde möchte den Fall eskalieren.
fuckin	nein	-	-4	Nur eine andere Schreibweise
lie	nein	-	-2	Vorher nicht vorhanden.
lying	nein	-	-2	Vorher nicht vorhanden.
freaking	nein	-	-3	Ähnlich wie fucking
omg	nein	-	-1	Vorher nicht vorhanden.
issue	nein	-	-1	Vorher nicht vorhanden.
seriously	nein	-	-1	Vorher nicht vorhanden.
mistakenly	nein	-	-2	Vorher nicht vorhanden.
screw	nein	-	-2	Vorher nicht vorhanden.
screwing	nein	-	-2	Vorher nicht vorhanden.
f*ck	Nein		-4	Nur eine andere Schreibweise
f**k	Nein		-4	Nur eine andere Schreibweise
f**king	Nein		-4	Nur eine andere Schreibweise
f***ing	Nein		-4	Nur eine andere Schreibweise
f*****g	Nein		-4	Nur eine andere Schreibweise
not	nein	-	-1	Wird häufig in einem negativen Kontext verwendet, daher mit einem niedrigen negativen Sentimentwert.
kind	ja	2	-	Wird zu 90% in Kombination mit of genutzt als kind of
damaged	nein	-	-3	Vorher nicht vorhanden.
scammed	nein	-	-2	Vorher nicht vorhanden.

TABELLE D.1: Änderungen am Sentimentlexikon - Fortsetzung

Wort	**in AFINN vorhanden?**	**Wert Vorher**	**Wert Nachher**	**Begründung**
big	ja	1	-	Kann nicht genau zugeordnet werden.
want	ja	1	-1	Meist negativ weil die Kunden ihre Ware wollen und noch nicht erhalten haben.
supposed	nein	-	-2	Meist negativ weil zB bestimmter Liefertermin nicht eingehalten wurde.
reach	ja	1	-1	Meist negativ weil die Kunden versuchen jemanden zuerreichen aber niemanden erreichen.
share	ja	1	-	Nur auf das Teilen von Infos bezogen daher neutral zu betrachten.
shares	ja	1	-	Nur auf das Teilen von Infos bezogen daher neutral zu betrachten.
shared	ja	1	-	Nur auf das Teilen von Infos bezogen daher neutral zu betrachten.
please	ja	1	-	Keine genaue Zuordnung möglich.
help	ja	2	-	Keine genaue Zuordnung möglich.
helping	ja	2	-	Ngram-Analyse hat gezeigt das ist in den meisten Fällen in Kombination mit not auftritt.
kidding	nein	-	-2	Vorher nicht vorhanden.
yes	ja	1	-	Hat keine Aussagekraft.

Anhang E

Elektronischer Datenträger

Eidesstattliche Erklärung

Ich versichere an Eides Statt, die von mir vorgelegte Arbeit selbständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen

Unterschrift: 

Ort, Datum: Bedburg, 30. Mai 2017